UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
CENTRO DE LETRAS E ARTES
ESCOLA DE BELAS ARTES
COMUNICAÇÃO VISUAL DESIGN

MARCELA MOTHÉ TELLES DE MIRANDA

METODOLOGIA E ATITUDE ÁGIL:
FUNDAMENTOS BÁSICOS PARA DESIGNERS UX

RIO DE JANEIRO
2019

Marcela Mothé Telles de Miranda

METODOLOGIA E ATITUDE ÁGIL: Fundamentos básicos para designers UX

Trabalho de Conclusão de Curso apresentado à Escola de Belas Artes da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do grau de bacharel em Comunicação Visual Design

Orientadora: Doris Kosminsky

Rio de Janeiro (Rio de Janeiro)
2019

Agradecimentos

Gostaria de agradecer por todo o carinho, compreensão e companheirismo de todos aqueles que me ajudam e ajudaram não só neste projeto, mas em todos momentos da minha vida.

Quero agradecer à minha mãe, Claudia, por ser meu maior modelo de força; aos meus avós, Maria Amélia e Luiz Antônio, por toda demonstração de carinho e conversas; ao meu avô Claudio por literalmente me dar meios de ir à faculdade e me levar até ela no início de tudo; ao meu tio Porphírio por pagar o Via Fácil; à minha avó Ana Maria que sempre compartilhou comigo seu gosto pelas artes; ao meu tio Luiz Gustavo que me ajudou a dar o pontapé inicial da minha carreira e por seguir me ajudando; à minha madrasta, Patrícia, pelo carinho e generosidade, e ao meu pai, Luiz Antônio Filho, que um dia em Arraial do Cabo virou para mim e disse "Filha, por que você não faz design?".

Quero agradecer ao meu namorado, Arthur Alesi, e à sua mãe, Edifrance, pela paciência, amor e carinho que tiveram por mim durante este período tão difícil.

Ainda quero agradecer aos meus amigos Gustavo Cardozo, Letícia Antunes, Luíza Freire, Victória Molgado e William Rabello pelas risadas, desabafos, festas, companheirismo, e inúmeros pitacos sobre design ao longo de todos estes anos. Sem vocês, nada disso seria possível.

À minha psicóloga, Cida, por me ajudar a processar toda sorte de emoções e seguir em frente.

Quero deixar toda minha apreciação aos dedicados professores do curso de Comunicação Visual Design. Em meio a tantas dificuldades, é incrível ver pessoas que realmente amam o que fazem e que conseguem mudar a vida de alunos.

Gostaria de agradecer à verdadeira estrela do curso de CVD, Kátia. Sem ela, muita gente não se formaria.

Gostaria de agradecer à minha orientadora, Doris Kosminsky, pelas lições e por me orientar nesta reta final.

E, por último, gostaria de agradecer a mim mesma. Foi difícil. Eu mereço esse biscoito.

Resumo

MIRANDA, Marcela Mothé Telles de. **Metodologia e Atitude Ágil:** Fundamentos básicos para designers UX. Rio de Janeiro, 2019.
Trabalho de Conclusão de Curso (graduação em Comunicação Visual Design)
Escola de Belas Artes, Universidade Federal do Rio de Janeiro, 2019

O presente projeto apresenta a proposta de design e conteúdo de um site, voltado para jovens designers interessados em metodologias ágeis, apresentando dicas e sugestões sobre como atuar dentro delas. Seu objetivo principal é explicar e tangibilizar conceitos abstratos e específicos das metodologias ágeis e da tecnologia, assim como dicas práticas a partir de ilustrações e texto explicativo. O projeto parte da reflexão sobre o papel do designer frente a novas tecnologias, especialmente no meio digital, e frente à relação com outros profissionais em um contexto de trabalho colaborativo. Para tanto, busquei referências teóricas em metodologias de trabalho Ágil e Waterfall, assim como, no Design Thinking a fim de produzir conteúdos práticos que ajudem a contribuir com os processos de designers UX atuantes, agindo de acordo com o que foi chamado de Atitude Ágil. Deste modo, este trabalho de conclusão de curso é principalmente um exercício teórico sobre métodos de design, somado ao projeto de um mínimo produto viável de um site, incluíndo com identidade visual e interface.

Palavras-chave: Metodologia Ágil, Design de Experiência, Design de Interface, Acessibilidade, Design Thinking.

Sumário

1. INTRODUÇÃO

Junto ao *boom da internet*[1] nas décadas de 1990 e 2000, surgiram uma série de métodos que propunham maneiras de otimizar o tempo e a performance de trabalho para criação de produtos digitais. Eles são baseados nos valores ágeis, um conjunto de princípios expressos pelo Manifesto Ágil, uma publicação online criada em 2001.

Este manifesto foi elaborado por um grupo de desenvolvedores de *software* e, em linhas gerais, propõe a adaptabilidade do projeto à diferentes condições como forma de se conseguir agilidade para a entrega de *software*. Em meio à competição por inovação tecnológica, seguir princípios ágeis pode ser uma vantagem.

Entretanto, esse manifesto foi feito considerando principalmente desenvolvedores como seu público alvo, deixando outros profissionais, como designers, sem saber muito bem com agir de acordo com suas diretrizes. Neste contexto, como designers podem contribuir dentro de metodologias ágeis? Como podem se tornar parte ativa da equipe? Como eles podem ser ágeis sem comprometer a qualidade da experiência em prol da entrega do *software*? Em um ambiente de trabalho que pode ser frequentemente caótico e que exige prontidão de resposta, é fácil se perder e se tornar reativo.

Este projeto tem por objetivo estudar e compreender as limitações e possibilidades de ação de designers UX dentro de modelos de trabalho ágeis de modo a esclarecer seus papéis e identificar maneiras pelas quais eles podem proceder de forma a realizar contribuições efetivas. Ao longo do texto, optei por preservar os jargões empregados no campo de UX (*User Experience*) em inglês, uma vez que estes são frequentemente utilizados desta maneira no próprio mercado de trabalho.

1.1. Objetivo Geral

Produzir e reunir em um site, dicas práticas sobre como designers podem contribuir dentro de modelos ágeis de trabalho, com foco em Scrum. Este site é dirigido para interessados e designers iniciantes nas metodologias ágeis.

---

[1] Expressão relacionada à popularização da internet

1.2. Objetivos Específicos

- Compreender como alinhar o papel do designer às condições de trabalho em modelos ágeis;

- Explicar e ilustrar os conceitos abstratos e complementares das metodologias ágeis a fim de deixá-los mais tangíveis;

- Oferecer dicas que contribuam com a qualidade de processos práticos e diminuam atritos durante a interação entre designers e desenvolvedores.

1.3. Justificativa

A importância deste projeto está em ajudar designers a agirem e a se reconhecerem como parte integrante de um complexo sistema formado por profissionais que possuem perspectivas distintas. Especialmente em métodos ágeis, o papel do designer é o de um comunicador e, semelhante aos outros integrantes, seu trabalho tanto afeta quanto é afetado pelo desempenho de outros profissionais.

Todos esses profissionais fazem parte de um sistema onde fluxos mal definidos de trabalho e desorganização custam caro para empresas e oneram trabalhadores. Estima-se que interações de baixa qualidade entre usuários e a intranet de empresas custam em média US$ 100 bilhões por ano[2], fora o tempo gasto em discussões diárias causadas pela falta de organização e ausência de um plano estratégico; as longas horas em reuniões; a saída de pessoas de empresas entre outras intempéries não previstas. Todos estes fatores fazem parte dos processos que constituem e afetam rotinas de trabalho.

Tanto a comunicação quanto a atitude destes profissionais perante os demais são fatores determinantes para o sucesso de um produto. O valor deste projeto está, portanto, em delinear o papel do designer e suas atitudes em confluência com o de outros profissionais frente às exigências que os produtos digitais e os métodos ágeis trouxeram. Realizar este trabalho é também ultrapassar a perspectiva do designer enquanto apenas comunicador visual e estendê-la ao papel do designer enquanto integrante de um sistema projetual.

---

[2] “Na verdade, eu estimo que a baixa usabilidade da intranet custa à economia mundial US $ 100 bilhões por ano em perda de produtividade dos funcionários (...) A boa arquitetura da informação torna os usuários menos alienados e suprimidos pela tecnologia.” (MORVILLE; ROSENFELD, 2006, p xi-xii)

> [...] A quarta área é o *design de sistemas ou ambientes complexos para viver, trabalhar, brincar e aprender*. (...) Esta área está cada vez mais preocupada em explorar o papel do design na sustentação, desenvolvimento e integração de seres humanos em ambientes ecológicos e culturais mais amplos, moldando esses ambientes quando desejáveis e possíveis ou adaptando-se a eles quando necessário. (...) Propriamente entendidas e usadas, estas áreas são também *lugares de invenção* compartilhados por designers, lugares onde uma pessoa descobre as dimensões do design thinking por reconsiderar problemas e soluções (BUCHANAN, 1992, p. 10, tradução nossa, grifo do autor)

1.4. Metodologia

Este trabalho é de natureza exploratória e a metodologia usada para conduzí-lo é *The Elements of User Experience* de Jesse James Garret. Seu método propõe um profundo entendimento sobre o problema como ponto de partida para a geração de soluções até a interface final. Ele se divide em cinco etapas que são: Estratégia, Escopo, Estrutura, Esqueleto e Superfície. Este método foi escolhido por comportar a Estratégia e os insumos gerados pela pesquisa exploratória como base para a construção das demais etapas.

A coleta dos dados durante a Estratégia foi feita através da consulta de materiais teóricos e por meio de entrevistas estruturadas e semi-estruturadas. Nos primeiros, foram estudadas especificidades de metodologias ágeis, com foco especial no Scrum, em contraponto ao tradicional método em cascata. Em seguida, de posse desses conhecimentos, foram elaborados questionários estruturados e semi-estruturados direcionados a designers e desenvolvedores, buscando coletar suas impressões sobre como se davam suas dinâmicas de trabalho e quais eram os pontos que provocavam fricção.

Desta etapa são gerados os insumos que pautam a definição da solução e de sua interface, compreendendo os demais passos de escopo (definição do que popula a interface), estrutura (como a interface é populada), esqueleto (a organização da interface) e superfície (onde aspectos visuais, interativos e navegacionais são definidos).

2. FUNDAMENTAÇÃO TEÓRICA

A seguinte porção da monografia discorre sobre os três principais pilares que sustentam a pesquisa: Uma breve introdução ao o que são as Metodologias Ágeis, com foco na vertente chamada *Scrum*; estas em contraponto ao método de trabalho *Waterfall*[3] e, por último, uma porção referente ao *Design Thinking* enquanto uma forma possível de *mindset* e metodologia para o trabalho de designers.

2.1. Metodologias de trabalho

2.1.1. Metodologias Ágeis

Em 2001, os integrantes originais da Aliança Ágil, representantes de diversas metodologias de desenvolvimento de software, se encontraram para discutir e uniformizar os princípios que os uniam sob a mesma perspectiva de *framework*[4] e desenvolvimento ágeis. Como resultado, surgiu o Manifesto para Desenvolvimento Ágil de Software, também conhecido como Manifesto Ágil. Os valores expressos por este manifesto também são resumidos usualmente pela palavra do inglês, *Agile*.

Este manifesto tenta romper com procedimentos burocráticos de trabalho que são considerados contrários à eficiência e é baseado nos princípios abaixo:

> "**Indivíduos e interações** mais que processos e ferramentas
> **Software em funcionamento** mais que documentação abrangente
> **Colaboração com o cliente** mais que negociação de contratos
> **Responder a mudanças** mais que seguir um plano
> Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda." (BECK, BEEDLE et al., 2001, grifo do autor)

Sob os princípios ágeis existe uma série de metodologias possíveis[5] com diferentes indicações dependentes das situações em que são aplicadas. Em última análise, depende da equipe definir e adaptar, dentre as metodologias existentes, a que melhor serve aos seus objetivos.

---

[3] No contexto de metodologias, é caracterizado por progressão sequencial e linear entre suas etapas

[4] "Estrutura de trabalho na modelagem de processos, uma estrutura de trabalho é qualquer associação planejada entre os modelos para atender uma política, desenho ou requisito de usabilidade." (HEFLO, [21-?])

[5] "O desenvolvimento ágil é uma estrutura conceitual e uma abordagem para o desenvolvimento de software com base nos princípios do Manifesto Ágil. O termo é um "guarda-chuva" para várias metodologias específicas baseadas em técnicas de desenvolvimento iterativo, nas quais requisitos e produtos evoluem através da colaboração entre equipes multifuncionais e auto-organizadas." (CPRIME, 2013, tradução nossa)

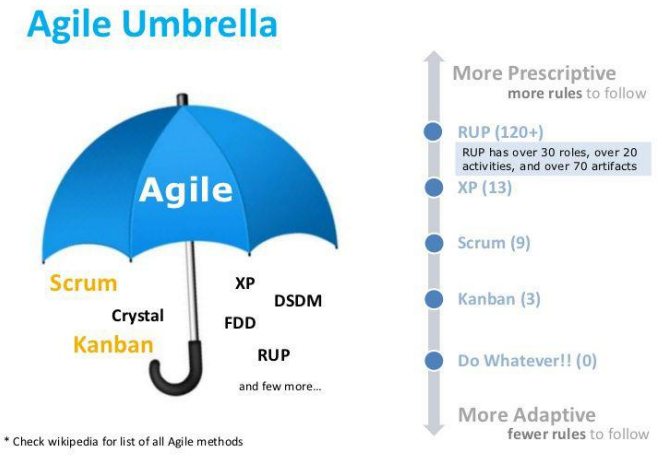


Figura 1 - Agile Umbrella (AMMOURI, 2015)

De maneira geral, as Metodologias Ágeis de trabalho se baseiam na divisão do projeto de um aplicativo ou *software* em pequenos blocos de funcionalidade. A construção, então, dá-se em partes através da soma entre antigos e novos blocos. Cada pedaço funcional entregue é testado (TAYMOR, [21---]) e, a partir do *feedback* [6] recebido, são feitos pequenos ajustes e alterações de modo a adequar a construção do *software* às novas necessidades em favor do cliente contratante ou mesmo para melhorar a aplicação entregue (BECK, BEEDLE et al., 2001).

O valor nesta "construção por partes" está na constante entrega de *software* funcional desde o início do projeto, permitindo que pequenos erros possam ser encontrados e reparados. A realização de testes é a maneira pela qual a Metodologia Ágil consegue assimilar a natureza imprevisível e mutável de um projeto dentro do seu *framework*, minimizando o impacto de imprevistos sobre a equipe de design e desenvolvimento, reduzindo, pelo menos em parte, cenários de retrabalho.

Metodologias Ágeis, portanto, não devem o termo "ágil" à velocidade de trabalho, mas sim à velocidade de detecção de problemas e ao tempo de resposta e adequação frente às mudanças e dificuldades; por isso, são consideradas extremamente adaptativas. Em outras palavras, suas qualidades de rapidez e adequação aparecem como resultados do fluxo de trabalho e não como requisitos para seu funcionamento.

[6] Resposta, reação. Neste caso específico de métricas

De acordo com um dos próprios membros da Aliança Ágil, Jim Highsmith (2001), o sucesso das metodologias ágeis está no compartilhamento de valores e princípios entre membros da equipe perante uma causa única:

> “No fundo, eu acredito que os metodologistas ágeis se baseiam em coisas “bregas” - [que elas são realmente pautadas em] entregar bons produtos para clientes por operar em um ambiente que faz mais do que dizer “pessoas são a nossa ferramenta mais importante”, mas que realmente “age” como se pessoas fossem o mais importante e que esquece da palavra “ferramenta”. Então, em última análise, a ascensão meteórica de interesse - e às vezes tremendas críticas a - metodologias ágeis é sobre a coisa brega de valores e cultura.” (HIGHSMITH, 2001, tradução nossa)

Modelos ágeis funcionam através de contato, convivência e prática, e, com isso, trabalham com a premissa de que a interação face-a-face e a qualidade dos processos são as maneiras mais eficientes para se atingir rapidez e adaptabilidade (BECK, BEEDLE et al., 2001). A maleabilidade do *workflow*[7] ágil como meio para atingir estes objetivos pode parecer contraproducente, mas faz sentido ao se pensar que este modelo busca criar pequenos sistemas dinâmicos em que “rapidez”, “coordenação” e “alinhamento” se retroalimentam e aumentam a produtividade da equipe por proporcionar à ela autonomia e autogestão como meios para eficiência. Essencialmente, isto quer dizer que a agilidade procura seguir processos, identificar seus erros e permitir que a equipe altere sua abordagem conforme o necessário.

> “O movimento Agile não é anti-metodologia. Na verdade, muitos de nós querem restaurar a credibilidade da palavra metodologia. Nós queremos restabelecer um equilíbrio. Nós aceitamos modelagem, mas não para que um diagrama seja arquivado em um repositório empoeirado. Nós aceitamos documentação, mas não centenas de páginas de volumes nunca ou raramente usados. Nós planejamos, mas reconhecemos os limites de se planejar em um ambiente turbulento. Aqueles que classificariam os proponentes de XP ou SCRUM ou qualquer uma das outras Metodologias Ágeis como "*hackers*" ignoram tanto as metodologias quanto a definição original do termo *hacker*.” (BECK, BEEDLE et al., 2001, tradução nossa)

Metodologias ágeis tendem a solucionar alguns dos problemas comumente encontrados em empresas como microgerenciamento, falta de comunicação e a sensação de falta de progresso (FORBES, 2014).

[7] Fluxo de trabalho

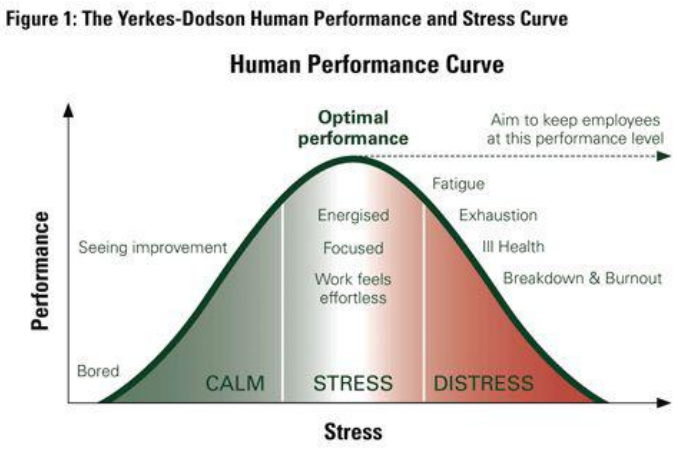


Figura 2 - The Yerkes-Dodson Human Performance and Stress Curve (CHEGG STUDY, [21--])

Outro ganho advindo da adoção do Agile é o embate natural, direto e constante entre o que é uma ideia ligada a um planejamento e o que é exequível. Como resultado, o gerenciamento de expectativas entre clientes e equipe se torna mais realista: há mais entendimento e transparência sobre o que de fato está acontecendo e o que pode ser feito e, com isso, há a atualização constante do plano estratégico para que ele atenda comercialmente ao cliente; o que é a maior prioridade da agilidade, como apontam seus princípios:

> **"Nossa maior prioridade é satisfazer o cliente** através da entrega antecipada e contínua de software valioso.
>
> **São bem-vindas mudanças nos requisitos, mesmo no final do desenvolvimento.** Os processos ágeis aproveitam as mudanças para a vantagem competitiva do cliente.
>
> Entrega de software de trabalho com frequência, de algumas semanas a alguns meses, com uma preferência pela menor escala de tempo.
>
> Empresários e desenvolvedores devem trabalhar juntos diariamente durante todo o projeto.
>
> Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte de que precisam e confie neles para realizar o trabalho.
>
> O método mais eficiente e eficaz de transmitir informações para e dentro de uma equipe de desenvolvimento é a conversa face a face.
>
> O software funcional é a principal medida de progresso.
>
> Processos ágeis promovem o desenvolvimento sustentável. **Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.**

> A atenção contínua à excelência técnica e ao bom design aumenta a agilidade.
>
> Simplicidade - a arte de maximizar a quantidade de trabalho não realizado - é essencial.
>
> As melhores arquiteturas, requisitos e projetos emergem de equipes auto-organizadas.
>
> Em intervalos regulares, a equipe reflete sobre como se tornar mais eficiente, depois tuna e ajusta seu comportamento de acordo." (BECK, BEEDLE et al., 2001, tradução nossa e grifo nosso)

Entretanto, é importante lembrar que os valores ágeis foram propostos por desenvolvedores e que eles dizem respeito à maneira como o *software* é desenvolvido. Apesar da maleabilidade de execução oferecida pelas diretrizes ágeis minimizar e ajudar a prever impactos, a estratégia de longo prazo depende fundamentalmente de organização e alinhamento entre *stakeholders*[8] sobre o que é este produto final e quais são seus indicadores de sucesso.

2.1.1.1. Scrum

O Scrum é a metodologia ágil mais popular e mais frequentemente aplicado a designers (LORANGER; LAUBHEIMER, 2019). Segundo um dos seus criadores, Jeff Sutherland, escritor do livro *Scrum: Como fazer o dobro do trabalho na metade do tempo*, sua experiência com gerenciamento de projetos até início dos anos 2000 usando *Waterfall*, também chamado método cascata, assim se sucedia:

> "Até aquele ponto — e até 2005 —, a maior parte do desenvolvimento de software era feita usando o método em cascata, no qual um projeto era concluído em todos os estágios distintos e seguia, passo a passo, em direção ao lançamento para os consumidores, ou usuários. O processo era lento, imprevisível e, em geral, nunca resultava em um produto que as pessoas queriam ou estavam dispostas a pagar para obter. Atrasos de meses ou até mesmo de anos eram endêmicos ao processo. Os planos iniciais de passo a passo, expostos em detalhes reconfortantes em diagramas de Gantt, asseguravam aos gestores que tínhamos total controle do processo de desenvolvimento — no entanto, quase sempre, nós rapidamente ficávamos atrasados em relação a eles, e desastrosamente acima do orçamento." (SUTHERLAND, 2014, tradução nossa)

[8] Público estratégico; todos aqueles que têm interesse, atuam ou são impactados em um projeto.

Criada em 1993, a metodologia SCRUM surgiu “para ser uma forma mais rápida, eficaz e confiável de criar softwares para o setor de tecnologia.” (SUTHERLAND, 2014).

> “O Scrum acolhe a incerteza e a criatividade. Coloca uma estrutura em volta do processo de aprendizagem, permitindo que as equipes avaliem o que já criaram e, o mais importante, de que forma o criaram. A estrutura do Scrum busca aproveitar a maneira como as equipes realmente trabalham, dando a elas as ferramentas para se auto-organizarem e, o mais importante, aprimorar rapidamente a velocidade e a qualidade de seu trabalho.” (SUTHERLAND, 2014, tradução nossa)

Para a redação do texto, equipe e time não são usados como sinônimos. “Equipe” se refere a todos os membros do Scrum, enquanto "time", é usado para designar aqueles responsáveis pela execução do produto.

Assim como outros *frameworks* ágeis, o Scrum é dividido em pequenos ciclos, as chamadas *sprints*, também conhecidas como iterações, que duram de uma a três semanas preenchidas por cerimônias como: *Planning*, *Daily* e *Review*[9].

Durante a reunião de *Planning* de um novo projeto, é feito um *kick-off*[10], uma reunião com a equipe[11] e o cliente voltada para o reconhecimento geral do que é o projeto e o que será feito. Nela são introduzidas informações como: objetivos de projeto; o time que participará; criação e priorização de funcionalidades no *backlog*[12] e o tempo de desenvolvimento para elas. Basicamente, seu objetivo é alinhar todas as partes integrantes sobre o que será feito e quando será feito.

Por privilegiar a satisfação do cliente, sua maneira de trabalho permite assimilar mudanças no projeto, por isso, dentre outras práticas que possibilitam esta maleabilidade, o escopo do projeto como um todo é fluido: requisitos podem ser abandonados ou aumentados, podem mudar de *sprint*, podem mudar de prioridade, etc. Para se estabelecer ordem e coesão, ao início de toda *sprint*, o escopo é fechado para que a equipe possa se dedicar àquelas tarefas e evitar que ela se

---

[9] Traduzido como “Revisão”, “Um relatório ou avaliação de um assunto ou eventos passados.” (OXFORD LIVING DICTIONARIES, [21-?])
[10] Traduzido como “Pontapé”. No caso, significa o ponto de partida do projeto.
[11] Mais adiante será explicada a dinâmica entre os membros de uma equipe de Scrum.
[12] Entendido como escopo do projeto, lista de *features* que será desenvolvidas.

desorganize, perca sua estratégia e, potencialmente, fique desmotivada. A curta duração das *sprints* e os constantes testes permitem que problemas sejam detectados cedo e que o *backlog* seja repriorizado caso necessário.

Com o *backlog* organizado, as *sprints* têm início. Durante este tempo, o projeto encontra-se em fase de execução. Esta fase é populada por práticas cotidianas de comunicação que aumentam a interface entre os integrantes da equipe e ajudam em sua auto-organização.

Uma dessas práticas é o *Daily*, uma reunião diária, idealmente no início do expediente, com as pessoas envolvidas no projeto. É uma reunião curta, com duração máxima de 15 minutos, dedicada à comunicar o status da tarefa de cada um. É opcional para equipe cronometrar sua duração ou a realizar de pé (*Daily standup*[13]), mas, diariamente, os membros da equipe devem responder às perguntas: “1. O que você fez ontem?; 2. O que você fará hoje? e 3. Você precisa de ajuda ou tem algum impedimento?" (TAYMOR, [21---], tradução nossa). Estas perguntas permitem a transparência sobre os status das tarefas, além da identificação rápida de problemas técnicos ou de dinâmicas de trabalho. Tendo conhecimento dessas informações, os membros da equipe podem agir sobre elas. Sendo este um momento de execução, é importante que a equipe se mantenha alinhada e dentro do fluxo de trabalho. Por isso, é aconselhável que tanto reuniões com o cliente quanto mudanças de *backlog* sejam evitadas para que o time possa se dedicar à entrega.

Ao final de cada *sprint*, é realizado o *Review*, uma reunião de revisão deste período conduzida com a equipe de projeto e os *stakeholders*. Nesta ocasião é apresentado para o cliente o que foi feito durante este tempo, como, por exemplo, *demos*[14] funcionais. Nesta reunião é possível fazer uma avaliação sobre como se sucederam as dinâmicas daquela iteração, mas algumas organizações preferem fazer uma reunião específica para este fim apenas com membros da equipe Scrum, chamada *Retrospectiva*. A partir da discussão sobre o que aconteceu na *sprint* anterior, é traçada a estratégia e as metas para este novo ciclo.

---

[13] Este método parte da premissa de que o desconforto as deixa mais eficientes e objetivas.
[14] Neste contexto, amostra de software

Até o final do projeto, estes procedimentos são repetidos, de modo a regularmente alinhar as partes integrantes.

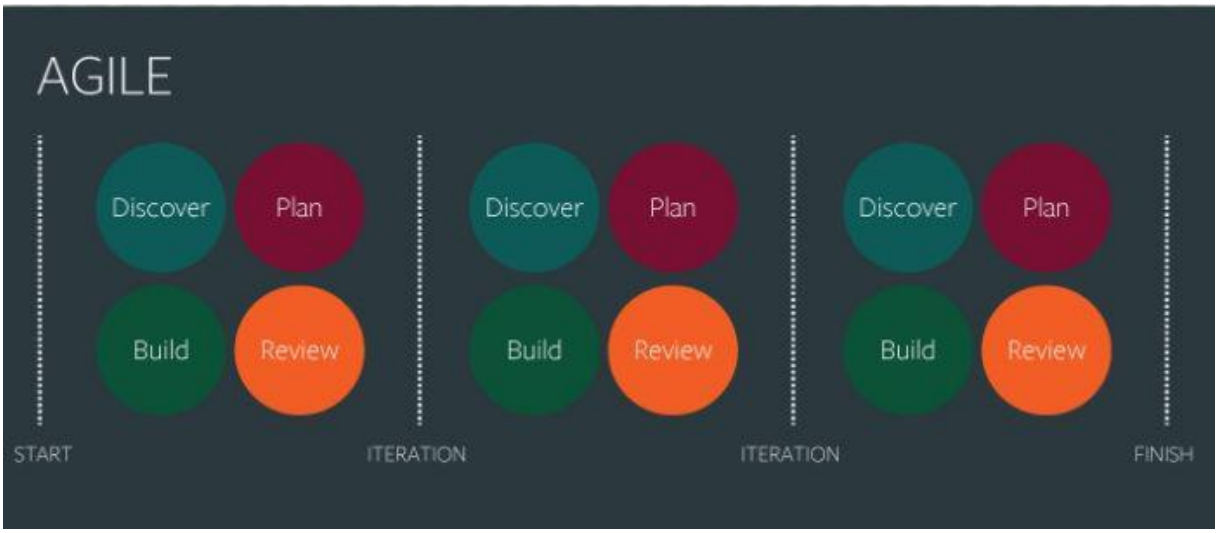


Figura 3 - Agile (NASH, 2019). Representação gráfica dos ciclos que compõem a organização e progressão de projeto em método ágil. Este método é marcado por pequenas entregas, repetições e revisões para que, ao longo do projeto, seus membros estejam sempre alinhados e cenários de erros sejam prevenidos e solucionados.

Para que os estes rituais sejam possíveis, é importante que a equipe como um todo esteja organizada e comprometida. Por isso, guiada pelos princípios consolidados pelo Manifesto Ágil, a estrutura de uma equipe Scrum é composta por figuras empenhadas em potencializar a eficiência de trabalho e do tempo de entrega:

- *Product Owner*: Responsável por trazer as demandas de *stakeholders* e clientes para a equipe[15]; por priorizar e gerenciar o *backlog*, traçar os objetivos da *sprint* e seu plano de *release*[16] e, com isso, determinar os custos do projeto. Basicamente, ele tem a tarefa de fazer com que todos entendam o que é demandado da sprint e é responsável pelos resultados dela.
  Para que o fluxo de trabalho funcione, é necessário que o *Product Owner* trabalhe em proximidade com o time e que ele esteja presente durante o *Planning* e *Review* da *sprint*. Ele deve respeitar a capacidade de absorção do time e sua auto-organização, intervindo só quando preciso.

[15] “No Scrum, o Product Owner tem autoridade final representando o interesse do cliente na priorização do backlog e nas questões de requisitos.” (COLLABNET, [21-?], tradução nossa)
[16] No caso, lançamento público, ao usuário final

- *Scrum Master*: É responsável por conectar e ser o facilitador entre o *Product Owner* e o time de *Scrum*. Dentre suas tarefas está fazer com que o time siga as regras e rituais da metodologia, sempre intermediando e facilitando esses processos com objetivo de potencializar a produtividade, e, com isso, atingir os objetivos da *sprint* apontados pelo *Product Owner*.
  Como o *Scrum* privilegia a auto-organização, o *Scrum Master* não gere o time, mas assiste sua produtividade e "(...) supervisiona o progresso da equipe durante o processo" (COLLABNET, [21-?], tradução nossa) através do recolhimento e exposição de informações sobre seu progresso para que *pain-points*[17] possam ser identificados e solucionados.

- Time de *Scrum*: O time é multifuncional, composto por engenheiros de software, designers UX, programadores, analistas de métricas, etc, com cerca de 8 profissionais que trabalham juntos e se comunicam diariamente.
  A partir da comunicação sobre quais são os objetivos da *sprint*, o time tem a autonomia e o dever de estimar o esforço de trabalho e seu prazo de entrega, assim como estabelecer suas dinâmicas com o apoio do *Scrum Master*.

É importante que os membros do time estejam comprometidos com o objetivo comum e que sigam as regras e rituais do *Scrum*. O time é visto como uma entidade e, por isso, sucessos e fracassos são compartilhados por todos os seus membros.

Por mais que a organização e as estratégias de ação entre os diferentes papéis existentes em uma equipe *Scrum* sejam pautadas em comunicação e contato cotidiano, as posturas e ações para contemplar as perspectivas dos diferentes profissionais do time de *Scrum* não é bem definida. Em nível de execução, a compreensão sobre as necessidades destes diferentes profissionais pode ser difícil, o que pode causar prejuízos no processo.

[17] Pode ser entendido como pontos de dificuldade (tradução nossa)

2.1.2. Waterfall

Waterfall é um método tradicional de gerenciamento de projetos baseado na segmentação clara e bem definida entre diferentes etapas de trabalho. Este modelo adota uma forma de progressão linear e sequencial, também sendo conhecido como método em cascata, em que a etapa subsequente é necessariamente dependente da etapa anterior.

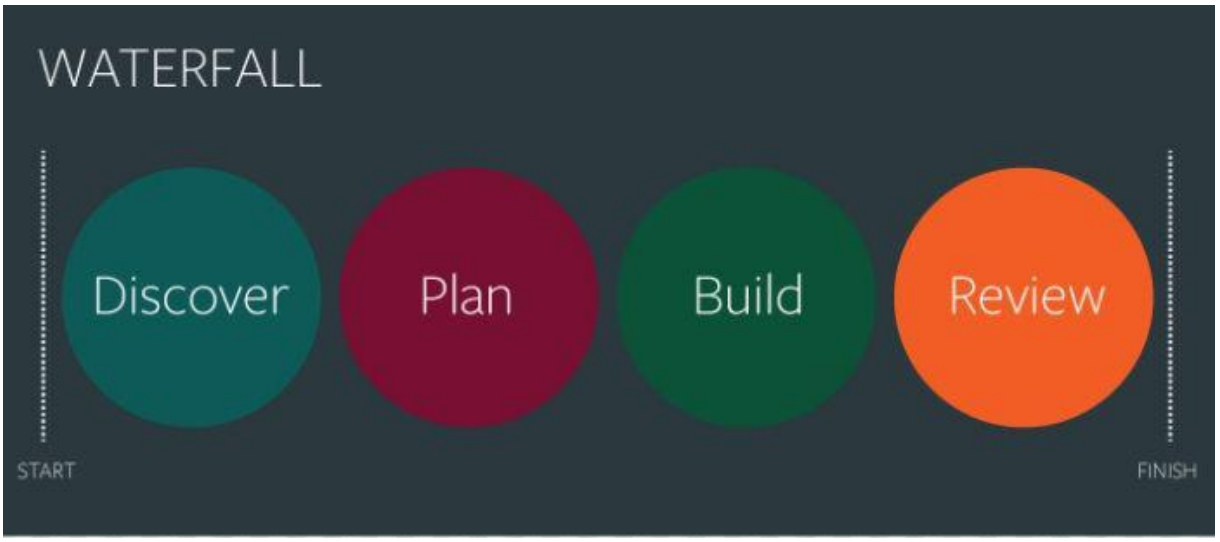


Figura 4 - Waterfall (NASH, 2019). Diferente da imagem associada ao método *Agile*, é possível ver que o plano estratégico em *Waterfall* é pautado na ideia de constante progressão “para frente” e no conceito de “encerramento” entre as fases e construção em cima de cada uma delas.

Considerando suas origens em processos manufaturados e industriais, o *mindset* aplicado a este modelo privilegia a minimização de risco entre uma fase e outra, o que, teoricamente, minimiza erros e surpresas com relação à performance do produto final e orçamento do projeto. Em contrapartida, isso envolve tornar o fluxo de trabalho um tanto engessado e protocolar, o que diminui a velocidade do projeto e suas possibilidades de experimentação.

> “O modelo de desenvolvimento em cascata originou-se nas indústrias de manufatura e construção; onde os ambientes físicos altamente estruturados significam que as mudanças no design se tornaram proibitivamente caras muito mais cedo no processo de desenvolvimento” (WATERFALL…, [21-?], tradução nossa)

Esse modelo é classicamente composto por 5 fases[18] (Requerimentos, Design, Implementação, Verificação e Manutenção[19]), com datas de início e fim definidas e com avanço de uma para outra condicionado à aprovação dos membros envolvidos. Ao fim de uma etapa, o estágio é reconhecido como "terminado", sem possibilidade de revisão ou qualquer tipo de retorno.

O estágio de Requerimentos é o que dá início ao projeto. Durante esta fase, são apontados os componentes do projeto; é uma forma abstrata de tangibilizar e documentar o que é o produto final em um conjunto rígido de requisitos que deve ser seguido e entregue. Analogamente ao *Scrum*, pode ser vista como a fase dedicada à apontar e organizar o escopo.

Após esta definição, a etapa de Design é dedicada à compreensão e tangibilização dos requisitos em desenhos para "(...) entender como que o produto final deveria se parecer" (SMARTSHEET, [21-?]). O que foi desenhado e, consequentemente, planejado é submetido à aprovação dos clientes para que todos os envolvidos concordem com o que será desenvolvido; até que isto aconteça, o desenvolvimento fica esperando as definições de Design.

Uma vez encerrado o estágio de Design, as *features*[20] desenhadas são passadas para a equipe de desenvolvimento para Implementação. Agora, tudo o que até então era "ideia" será desenvolvido e concretizado em *software* funcional.

Uma vez que tudo está feito, o *software* deve passar pela etapa de Verificação, quando se valida aquilo que foi desenvolvido para aprovação. É uma oportunidade para condução de testes para descoberta de *bugs*[21] e verificação da qualidade do código e serviço. Geralmente, uma reunião de *UAT* (*User Acceptance Test*, com tradução nossa para Teste de Aceitação de Usuário) é conduzida com a equipe de desenvolvimento, designers e clientes antes do *release* para público.

---

[18] Algumas fontes variam e apontam entre 5, 6 ou 7 fases. Para redação, considero o agrupamento de algumas dessas fases em uma única por questão de semelhança entre seus processos, não sendo proveitoso exibir suas granularidades.

[19] Nomenclatura em função de grande parte dos diagramas encontrados usarem estes termos.

[20] Funcionalidades do software

[21] Termo comum em desenvolvimento, pode ser traduzido como um "defeito" que impede que uma aplicação se comporte como esperado. Apesar de sua tradução literal ser "inseto", o termo ficou conhecido como "defeito" em informática em função de uma mariposa que entrou em um computador e impediu seu funcionamento. A situação se passou e foi registrada por Grace Hooper, programadora da Marinha dos EUA em 1947. (FETTER, 2014)

Uma vez lançado, a etapa de Manutenção foca no código para corrigir erros, fazer melhorias e, a partir do *feedback* de usuários reais, consertar *bugs* reportados.

Waterfall é um modelo que é melhor aplicado a projetos com pouca possibilidade de mudança de requisitos, isto é, que não sejam dinâmicos ou de caráter experimental (PROJECT BUILDER, 2017). É uma forma de gestão que deixa claro como se sucederão as fases de projeto, porque "Tradicionalmente, a gerência quer duas coisas: controle e previsibilidade" (SUTHERLAND, 2014, p. 12). O problema é que, em desenvolvimento de *software*, dependendo do objetivo do produto e da qualidade do código, talvez não seja possível entregar o que foi planejado. Como isso é possível se todo o planejamento tinha sido traçado?

> (...) em vez de descartar o plano ou o modo como pensam nele, os gerentes contratam pessoas para fazer com que pareça que o plano está funcionando. (...) [Por isso,] **os relatórios se tornaram mais importantes do que a realidade que deveriam descrever, e se houvesse discrepâncias, o problema estava na realidade, não nos diagramas** (SUTHERLAND, 2014, p. 9, tradução nossa, grifo nosso).

2.1.3. Comparativo *Agile* x *Waterfall*

A grande variedade de metodologias de trabalho (*Agile*, *Waterfall*, entre outras), justifica-se pela necessidade de diferentes abordagens de gestão e organização para contemplarem as especificidades de variados projetos. A escolha pelo método deve ser feita através da compreensão dos objetivos, natureza e do contexto de projeto. Neste sentido, cada projeto se adequa melhor à uma metodologia. Os modelos *Waterfall* e *Agile* têm propostas fundamentalmente distintas e, por isso, cada um possui suas vantagens, desvantagens, e, principalmente, contextos de aplicação distintos.

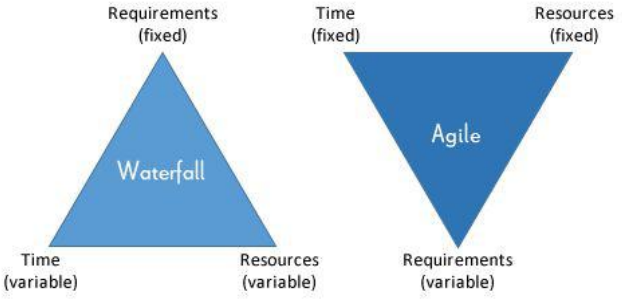


Figura 5 - Valores e prioridades Waterfall x Agile. (SCHAEFFER, [21--])

Waterfall proporciona uma visão clara sobre o futuro e um caminho simples (linear e sequencial) para se chegar até o desejado, por isso, é visto como uma maneira simples de se planejar e contemplar etapas. Como ilustrado pela figura acima, enquanto o *Waterfall* tem seus requerimentos "fixos", isto é, tem um escopo fechado e uma definição clara do que *deve ser* entregue ao final; o *Agile* lida com requerimentos de maneira flexível e adaptativa às *condições*. Dependendo do tipo de projeto, estas podem ser características-chave para se alcançar os objetivos de negócio e vantagens competitivas, uma vez que a flexibilidade ágil permite testes, pequenos *releases* e retorno imediato, o que é interessante para um projeto que consiste em uma plataforma de publicação de site, mas que seria de alto risco para a criação de um *software* de monitoramento de pacientes, por exemplo.

A escolha por estas duas formas de gestão foi proposital para a elaboração de uma análise comparativa entre suas respectivas maneiras de funcionamento e valores. O resumo dessas qualidades está na tabela a seguir:

	Scrum (metodologia ágil)	*Waterfall*
Progressão do projeto	Iterações cíclicas e incrementos	Linear e sequencial
Vantagens	Testes, aprendizado e adequação a cada sprint	Minimização de riscos
Orçamento	Mutável	Fixo
Requisitos	Mutáveis	Fixos
Equipe	Mesma equipe do início ao final	Rotatividade de membros da equipe
Interações de trabalho	Comunicação cara-a-cara	Protocolos e processos
Maneira de trabalho	Autonomia da equipe, horizontal	Estrutura *top-down*[22], vertical
Estratégia	Adaptação aos cenários	Previsibilidade e planejamento

Tabela 1 - Tabela comparativa Scrum (metodologia ágil) x *Waterfall*

[22] Sinônimo de para estrutura empresarial vertical, onde as decisões vem de uma esfera de gestão (top) para o nível de execução (down)

Os objetivos comerciais têm mais chance de serem atingidos, uma vez que o entendimento sobre o problema e desenho da solução sejam feitos para posteriormente serem executados em código. É nesta fase essencialmente ligada à estratégia de produto que os designers UX entram em cena. Dentre um de seus métodos de trabalho mais comuns está o *Design Thinking*, uma abordagem centrada no entendimento dos desejos e necessidades dos usuários. Como este modo de pensar, que contempla descobrir e delinear o problema, pensar a estratégia, pensar nos *touchpoints*[23] de produto e na elaboração do artefato final, pode se adequar a ambientes ágeis onde problemas técnicos e repriorização de *backlog* são comuns?

2.2. Design Thinking

Design Thinking é um processo de geração de soluções que concilia viabilidade técnica, objetivos de negócio e necessidades do usuário. É justamente a união destes três elementos, enquanto constantes tanto para o fluxo criativo quanto para criação da solução, que atribuem o diferencial da essa maneira de pensar.

Ele propõe soluções centradas no usuário final, e é conhecido como *Human-Centered Design*. Por isso, suas práticas para elaboração de soluções giram em torno da "(...) habilidade humana de ser intuitivo, de reconhecer padrões e de construir ideias que são emocionalmente tanto significantes como funcionais" (IDEOU, [21-?], tradução nossa).

Este processo sugere o uso de uma série de etapas de investigação e criação que emulam o método duplo diamante, onde o processo criativo oscila entre a geração de opções que divergem e a definição de caminhos que convergem (ZABAN, 2018) a fim de delinear o problema, explorá-lo e definir uma solução. Etapas como pesquisa, definição, ideação, prototipação e teste fazem parte do processo, na busca pelo entendimento do problema e do usuário como ponto focal. Nelas são produzidos os materiais, também chamados de entregáveis, que transmitem a proposta de solução.

[23] Ponto de contato. Qualquer momento de interação de um consumidor com uma empresa

Dentro do contexto de empresas, anterior à investigação do problema estão os objetivos de negócio como propulsores do processo criativo. É possível que o objetivo de negócio não seja necessariamente *o* problema a ser resolvido ou que ele mesmo vá contra ao que é considerado como o melhor para o usuário; o que remete diretamente a qualidade *"wicked"*[24] de muitos problemas de design.

> (...) a resposta para esta pergunta está em algo raramente considerado: a natureza peculiar do design. Os problemas de design são "indeterminados" e "*wicked*", porque o design não tem um objeto especial para si próprio além daquele que um designer concebe como sendo [em um projeto]. (...) Mas no processo de aplicação, o designer deve descobrir ou inventar um *determinado* objeto a partir dos problemas e questões de circunstâncias específicas. (BUCHANAN, 1992, p. 16, tradução nossa)

O design não tem uma questão em si. Ele serve enquanto ferramenta para se alcançar outros objetivos; daí a importância da sua
relação com os outros dois pilares: a viabilidade técnica e objetivos de negócio.

> No final do século XIX e início do século XX, o design gráfico foi orientado para a expressão pessoal através da criação de imagens. Foi uma extensão da expressividade das belas artes, pressionada para o serviço comercial ou científico. Isso foi modificado sob a influência da "teoria da comunicação" e da semiótica quando o papel do designer gráfico foi transferido para o papel de um intérprete de mensagens. (BUCHANAN, 1992, ep. 11, tradução nossa)

O que leva à redefinição do papel do designer e de sua postura perante outras figuras do processo. O designer é um mediador que deve colaborar na busca pelo entendimento do problema e do usuário como ponto focal.

[24] No contexto do Design Thinking, significa problemas com má formulação, mal definidos. “Rittel argumenta que a maioria dos problemas endereçados aos designers são problemas ruins (wicked problems) […] uma ‘classe de problemas sociais mal-formulados, onde a informação é confusa, onde existem muitos clientes e a tomada de decisão possui conflitos de valores, e onde as ramificações em todo o sistema são completamente confusas’.” (BUCHANAN, 1992, p. 15)

2.3. Conclusões a partir das leituras

O equilíbrio entre o modo de pensamento de designers e o faseamento de execução proposto pelos valores ágeis é difícil de ser atingido. Em seu artigo *Are agile methods good for design?*, John Armitage, co-fundador e diretor de investimentos da Egerton Capital, uma empresa de fundos de cobertura, pontua a fundamental diferença entre eles:

> Os métodos ágeis buscam se beneficiar da inteligência de experimentar a existência real do produto, e quanto mais cedo melhor. Design, por outro lado, visa prever o que todo o produto será antes que ele exista. (ARMITAGE, 2004)

Soma-se a isso, o entendimento da autora de que nos materiais produzidos por designers, chamados documentação, ainda estavam as informações fundamentais para a concretização da solução do projeto.

Com isso, a hipótese inicial deste projeto girou em torno da ideia de que a passagem de informações entre designers e desenvolvedores era um problema e que este poderia ser suavizado pela transmissão e alteração rápida de dados conforme as necessidades do projeto pela automatização deste processo.

Ao mesmo tempo, através de leituras, foi pressuposto que o principal ponto negativo estaria na passagem de informações técnicas sobre a navegação do sistema. Por isso, a primeira proposta de solução era a criação de uma ferramenta que automatizasse a criação destes fluxos através da adoção do vocabulário visual de James Garrett[25], cujas formas teriam condicionais reais de código para exportação e uso de desenvolvedores.

[25] Um conjunto de formas e símbolos padronizados criados pelo designer Jesse James Garrett para construção de fluxos de navegação

3. DESENVOLVIMENTO

3.1. Entrevistas e seus objetivos

Como é apontado nas boas práticas do *Design Thinking*, a imersão em um problema passa por uma etapa de pesquisa para que designers sejam capazes de contemplar a questão sob o ponto de vista do usuário, com objetivo de produzir uma solução com valor efetivo. Por isso, para além do levantamento teórico, foram utilizados dois tipos diferentes de pesquisa exploratória[26]- entrevistas estruturadas e semi-estruturadas.

> Por fim, você deve considerar qualquer método que possa expô-lo às declarações diretas dos usuários de suas próprias necessidades e, quando puder, use uma combinação de métodos para abranger o maior número possível de bases. (MORVILLE; ROSENFELD, 2006, p.38, tradução nossa)

Os dois tipos de pesquisa selecionados fornecem diferentes informações mas são complementares, uma vez que entrevistas estruturadas fornecem dados quantitativos e entrevistas semi-estruturadas, dados qualitativos (THE TRIANGLE ADMIN, 2016). A mistura entre dados *quali* e *quanti* permite melhor leitura e entendimento sobre a questão investigada.

Através do conhecimento adquirido pela pesquisa teórica, a construção dos roteiros girou em torno de perguntas relacionados às práticas e impressões sobre o *workflow* dos respectivos entrevistados.

Para isso, independente do tipo de entrevista, o roteiro foi essencialmente formado por questões que buscavam entender quem eram essas pessoas (o que elas fazem e há quanto tempo); quais informações são importantes para o trabalho delas; quais etapas formam o trabalho delas; quais suas percepções e vivências de seus fluxos de trabalho; como informações são tratadas e transmitidas; qual o ciclo de vida dessas informações; quem são os outros agentes do ecossistema de trabalho; quais ferramentas elas usam; o que elas entendem como documentação e quais questões técnicas e problemas interferem no andamento do projeto.

---

[26] "A **pesquisa exploratória** tem o objetivo de trazer um maior entendimento sobre o objeto da pesquisa, seja ele um fato ou um fenômeno (...)" (MARTINS, 2017)

O público-alvo para as entrevistas consistia principalmente de designers e desenvolvedores com algum tipo de experiência de trabalho. Estas entrevistas são analisadas mais adiante.

3.1.1. Entrevistas estruturadas

Entrevistas estruturadas são fundamentalmente compostas por perguntas diretas que limitam as variações de respostas. Sua abordagem objetiva foi o ponto determinante para a escolha de sua distribuição online.

Durante a elaboração da entrevista, o principal desafio foi não onerar os entrevistados. Por isso, o questionário foi montado com o menor número possível de perguntas, sempre buscando obter dados ricos de informação de maneira objetiva. Para isso, foram necessários diversos processos de lapidação, que levaram muito tempo, transformando este conteúdo denso e cheio de variáveis em opções objetivas para seleção.

A autora partiu do princípio de que a distribuição deste material via redes sociais (por grupos de WhatsApp e Facebook) e por pedidos diretos a pessoas acabaria limitando a amostra de pesquisa ao seu grupo social: universitários ou recém-formados com pouca experiência no mercado de trabalho.

No final, o questionário como um todo é formado por três roteiros, um para designers UX, um para desenvolvedores e, um último para quem, ao mesmo tempo, desempenha ambos papéis, ultrapassando o público a princípio previsto para as entrevistas. Cada roteiro tinha em média 18 perguntas, 15 de múltipla-escolha obrigatórias e 3 discursivas facultativas (Apêndice A). Feitos pela ferramenta Formulários Google, o questionário é formado por diversos pontos de decisão que levam o usuário a certas perguntas e omitem outras conforme suas respostas. A amostra mínima era de 15 respostas, mas foram obtidas 17.

Tanto os roteiros das entrevistas estruturadas quanto os dados extraídos a partir delas estão disponibilizados na seção dedicada aos *Apêndices* da monografia.

3.1.2. Entrevistas semi-estruturadas

Entrevistas semi-estruturadas permitem maior maleabilidade sobre seu conteúdo. Seu valor está na possibilidade de discutir e entender a linha de raciocínio dos entrevistados e direcionar, aprofundar ou evitar certos temas. Como contraponto ao questionário online, as entrevistas semi-estruturadas foram direcionadas a profissionais mais experientes do círculo de convivência profissional da autora.

Como os entrevistados foram selecionados antes da construção do roteiro, as perguntas foram feitas pensando nos pontos fortes de cada um deles, antecipando possíveis "caminhos" da conversa e mergulhando em temas que eles dominavam. Ao total, os roteiros desenvolvidos, um voltado para designers e outro para desenvolvedores, tinham em torno de 20 perguntas cada.

Neles, a autora teve a oportunidade de explorar mais as respostas que recebia. O questionário foi conduzido conforme o que parecia ser melhor: por vezes perguntas foram omitidas, criadas, ajustadas e "condensadas" em uma mesma. No final, o roteiro se comportou mais como um guia de ideias e diretrizes para conversa do que propriamente como uma listagem sequencial de questões.

Foram entrevistados 2 desenvolvedores e 2 designers, com uma média de 10 anos de carreira cada. Pela conversa, foi possível computar mais variáveis que interferem durante as interações entre membros de um time *Scrum*. O saldo destas entrevistas foi extremamente positivo, uma vez que *insights* diferentes foram extraídos e perspectivas diferentes foram contempladas.

As entrevistas foram vis-à-vis e em sua maioria presenciais, com exceção de uma feita via videoconferência. A estipulação de uma hora de duração para elas viabilizou uma postura exploratória, quase que como uma conversa informal.

Os roteiros das entrevistas semi-estruturadas e suas respectivas transcrições estão na seção dedicada aos *Apêndices* da monografia. Nas citações a seguir e na transcrição, os designers estão nomeados como UX1 e UX2 e os desenvolvedores estão nomeados Dev1 e Dev2.

3.1.3. Descobertas das entrevistas

O que inicialmente acreditava-se ser um problema relacionado à *qualidade técnica* de artefatos, na verdade, revelou-se após as entrevistas como dois problemas distintos: por um lado, a *dificuldade comunicativa* entre membros do time de desenvolvimento e design e, por outro, a dificuldade em acompanhar e lapidar as soluções constantemente, uma tarefa necessária para os designers.

> “E daí, o que acontece muitas vezes, pelo time de desenvolvimento estar desenvolvendo de maneira interativa e incremental, bota na rua as coisas, coleta feedback e tem que voltar para a figura de UX que está distante e dizer "Cara, isso aqui mudou" e daí (a figura de UX responde) "Não, mas eu pensei assim. O meu design está feito". Por isso, esse ciclo é importante. Por isso que eu acho que com o profissional "tando junto", ele enxerga esse ciclo fechando e daí se torna mais aberto, mais receptivo; e daí, com isso também, a documentação, ela é verdadeira, digamos assim.” (Dev1, 2019)

Uma vez que a mentalidade *Fail fast, learn faster*[27] é o que possibilita o aprendizado e a lapidação de uma solução a partir de dados reais, o design feito está constantemente suscetível à mudanças. A vida de arquivos e as soluções em *Agile*, portanto, estão sujeitas à constante reavaliação: se determinada solução continua a se encaixar no projeto, ou ainda se sua criação é sequer viável de acordo com os desenvolvedores naquele momento. Em modelos ágeis de trabalho, estar alinhado e aberto à discussão com os demais membros do time é essencial. A documentação vem apenas como consequência deste *workflow*. "Comunicar" em *Agile* é qualquer interface que permita a diálogo e a troca de ideia: "Eu acho que, uma, uma coisa que é inerente à documentação, que para mim, é sempre um problema, é receber um pedaço de papel, seja ele físico ou eletrônico e não ter uma conversa em cima dele." (Dev 1, 2019).

Uma vez que o diálogo entre os membros do time é a base para o funcionamento da metodologia *Agile*, a produção de artefatos visuais muitas vezes é dispensável: eles são feitos na medida do possível e podem assumir diferentes formas. Não existe um conjunto de artefatos padrão para chamar de documentação. Documentação é todo registro de projeto.

---

[27] Traduzido como "Falhe rápido, aprenda mais rápido", é uma expressão que transmite o princípio ágil de adaptabilidade e aprendizado frente à experimentação. É uma expressão que "normaliza" o erro para se chegar ao acerto através da experiência factual.

> Eu acho que na hora que tu tá entendendo o problema, fazendo tuas pesquisas para entender o problema, entrevistando as pessoas, teus stakeholders para entender o problema real como ele é, muitas vezes, tu vai acabar rafeando, rascunhando coisas para tentar chegar a um entendimento, aquilo ali é um documento, é um rascunho. A partir do momento em que isso se torna o código, se torna o artefato em si, aquela tua foto da parede perdeu o sentido de existir, porque tu já tá com aquilo pronto, já tá com aquilo documentado (Designer 2, 2019)

Por isso, talvez seja necessário distinguir o que é documentação do que é estratégia de acordo com a metodologia de trabalho. Para *Waterfall*, a documentação parece ser o mesmo que a estratégia, no sentido de que é registro da visão final do projeto; a maneira de execução é então organizada de acordo com os recursos e é feito um cronograma específico para se atingir esta visão concretizada do que é o produto final. Para *Agile*, em contrapartida, a visão final do produto é mais fluída e volátil justamente por ser um modelo de trabalho que permite experimentação: o que está no código é a documentação funcional do momento atual, entretanto, é preciso entender que ele não é necessariamente o objetivo final do projeto.

Entre o código e a visão de projeto existem inúmeras lacunas, remendos e reformulações, e é preciso que designers e desenvolvedores se entendam para que algo de valor possa ser entregue. Logo, é necessário que exista um planejamento estratégico com foco na experiência para que cada entrega de *software* funcional seja uma entrega de efetivo valor para o usuário final e, por consequência, uma entrega de valor para o cliente.

3.2. Definição do problema

Através das entrevistas, ficou claro que muitos designers não entendem como atuar em metodologias ágeis: seu modo de pensar não se adequa a este modo de execução. Como é que designers que são ensinados desde sempre a pensar no todo e a ter a visão do produto final podem se adequar a um método de trabalho que exige que eles pensem por partes e que entendam que nem sempre o que foi projetado é tecnicamente viável?

É interessante pontuar que, apesar da própria natureza do design ser tão intrinsecamente ligada à tecnologia (MOURA, [21--]), o problema em encontrar o equilíbrio entre ideação e execução é algo que já foi apontado por Buchanan (1992) bem antes do surgimento do Manifesto Ágil.

> Em vez de produzir interações produtivas, o resultado é muitas vezes confusão e quebra de comunicação, com uma **falta de prática inteligente para levar idéias inovadoras à incorporação objetiva e concreta**(...). Sem uma reflexão adequada para ajudar a esclarecer a base da comunicação entre todos os participantes, há pouca esperança de compreender os fundamentos e valores do design thinking em uma cultura tecnológica cada vez mais complexa. (BUCHANAN, 1992, ep. 8, tradução nossa, grifo nosso)

Diretrizes e processos relacionados à como agir de acordo com os princípios ágeis foram essencialmente construídos sob a ótica do desenvolvimento de *software*. Mesmo a busca por diagramas explicativos sobre o funcionamento das *sprints* do Scrum, dificilmente localizam o trabalho de design dentro do processo.

O problema, portanto, está no desconhecimento sobre quais atitudes práticas o designer pode ter para contribuir com o time e qual é o seu papel frente aos outros membros do projeto e este problema já havia sido .

Por consequência, a solução está em buscar maneiras de conciliar as expectativas e necessidades das metodologias ágeis com o *mindset* e as atitudes de designers a este processo por meio da proposta de dinâmicas de trabalho que melhorem a qualidade do design e que assimilem a volatilidade do projeto sem a qualidade do design ser prejudicada. Se o objetivo é melhorar os produtos, é fundamental melhorar a qualidade dos processos e da maneira de projetar.

3.3. Proposta

A abordagem adotada é a criação do design de um *website* que contenha dicas e diagramas, explicando de maneira simples o papel do designer UX, o que são os princípios ágeis e, como unir os processos de design aos processos ágeis por meio de atitudes práticas.

Esta escolha se dá pelo popularidade de artigos sobre UX, formato popularizado por grandes portais como Medium e Norman Nielsen Group, e pelo fácil e democrático acesso que este conteúdo teria, além da possibilidade de usar recursos visuais como ilustrações e animações para representar conceitos abstratos.

É importante pontuar os limites desta solução. Considerando que a raiz do problema é processual, o sucesso do produto poderia até ser estimado por métricas como número de visitas ao site, tempo na página, compartilhamentos, etc, porém, o real valor dele estaria no aprendizado e na prática de suas diretrizes no cotidiano de equipes de ágeis. É até mesmo por esse motivo que outras abordagens possíveis como a elaboração de um curso ou palestra não foram contempladas: porque este tipo de problema seria solucionado por meio do exercício prático.

3.3.1. Definição do público alvo

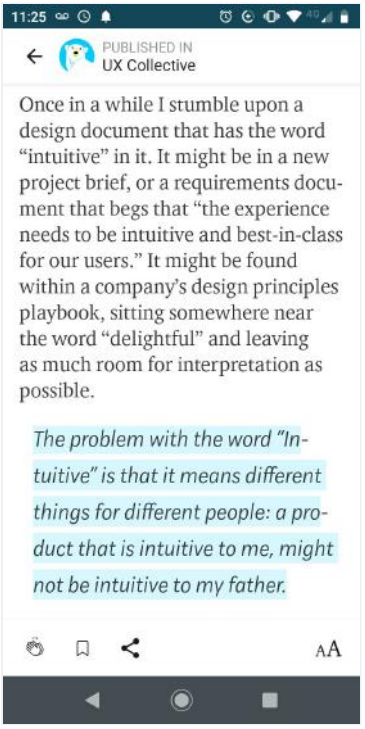
O site é primariamente destinado a designers UX que não estão familiarizados com metodologias ágeis e que não sabem como agir dentro de ambientes de trabalho que as adotam. O objetivo é explicar de maneira simples atitudes ágeis que designers podem ter, exemplificadas pelo Scrum, assim como compartilhar boas práticas técnicas para criação e organização considerando o meio digital.

3.4. Benchmarking

A pesquisa por referências girou em torno principalmente da análise sobre como grandes sites voltados a artigos sobre design exibem seu conteúdo. Como parâmetros para análise estabeleci: construção de texto, uso de recursos tipográficos, exibição de imagens e navegação pelo conteúdo.

3.4.1. Medium

No caso do Medium, analisei seu aplicativo móvel (*mobile*[28]). A versão *app* oferece um *bottom app bar*[29] que contém ações como dar "*like*", salvar o artigo, compartilhamento e ajuste do tamanho de fonte. Considerando que o Medium é uma plataforma e que a proposta deste projeto é um *site stand-alone*[30], as funções de salvar e favoritar não se aplicam. Além disso, navegadores pelos quais o *site* seria acessado já suportam as funções de compartilhamento e ajuste de fonte.



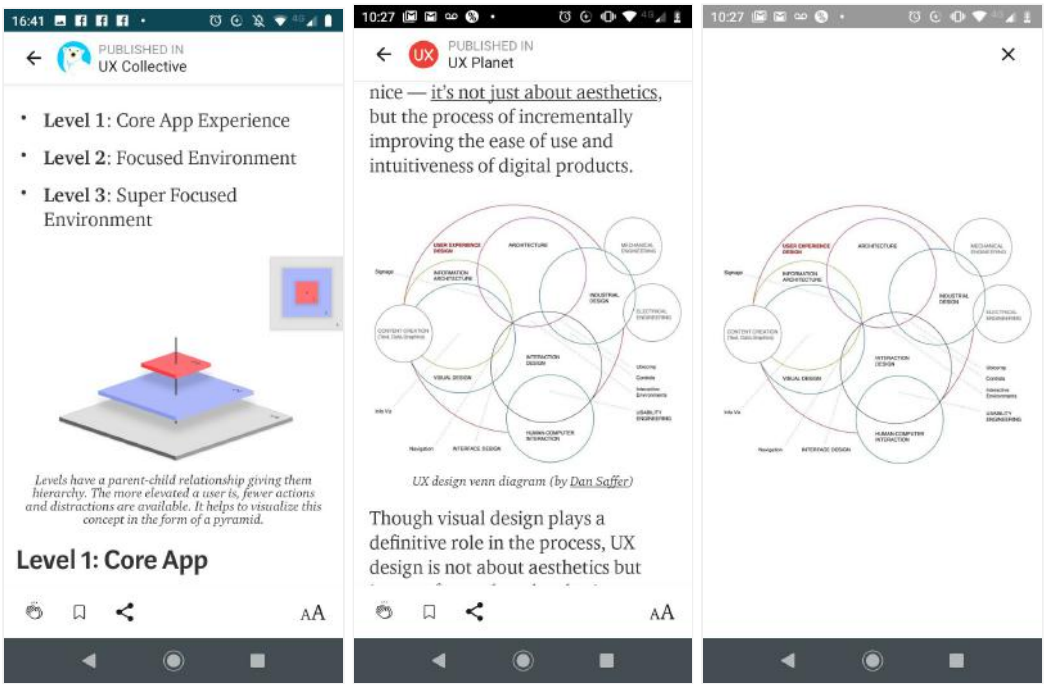
Figuras 6 e 7 - *Prints*[31] do aplicativo Medium

Existe pouca variação tipográfica; a diferenciação se dá mais por meio de espaços brancos, pelo aumento do tamanho da tipografia e uso de itálico para destaque, contrastando com o peso regular que compõe a mancha de texto. As porções do texto variam entre título, texto corrido, olho do texto e tópicos. Uma funcionalidade interessante quanto ao olho de texto é que, por se tratar de uma frase de destaque, o *app* tem a opção de copiá-lo para compartilhamento.

---

[28] Dispositivo celular portátil
[29] Barra inferior de aplicativos que fornecem acesso a links de navegação. Idealmente, esta barra é composta minimamente por 3 links e, no máximo, por 5.
[30] Site autossuficiente
[31] No contexto, captura da imagem em tela

Ao longo do texto, também existem *hiperlinks* que levam para outros artigos dentro do Medium. Quanto às imagens, geralmente esquemas e fluxogramas são usados para ilustrar os textos. É interessante notar que é possível visualizar uma imagem em *full screen*[32] aberta em um modal[33], apesar de não existir nenhum tipo de pista visual deste recurso.



Figuras 8, 9 e 10 - *Prints* do aplicativo Medium

Estas, porém, não recebem nenhum tratamento para consumo via *mobile*, o que é especialmente prejudicial quando elas têm muitos detalhes ou são horizontais.

Figuras 11 e 12 - *Prints* do aplicativo Medium

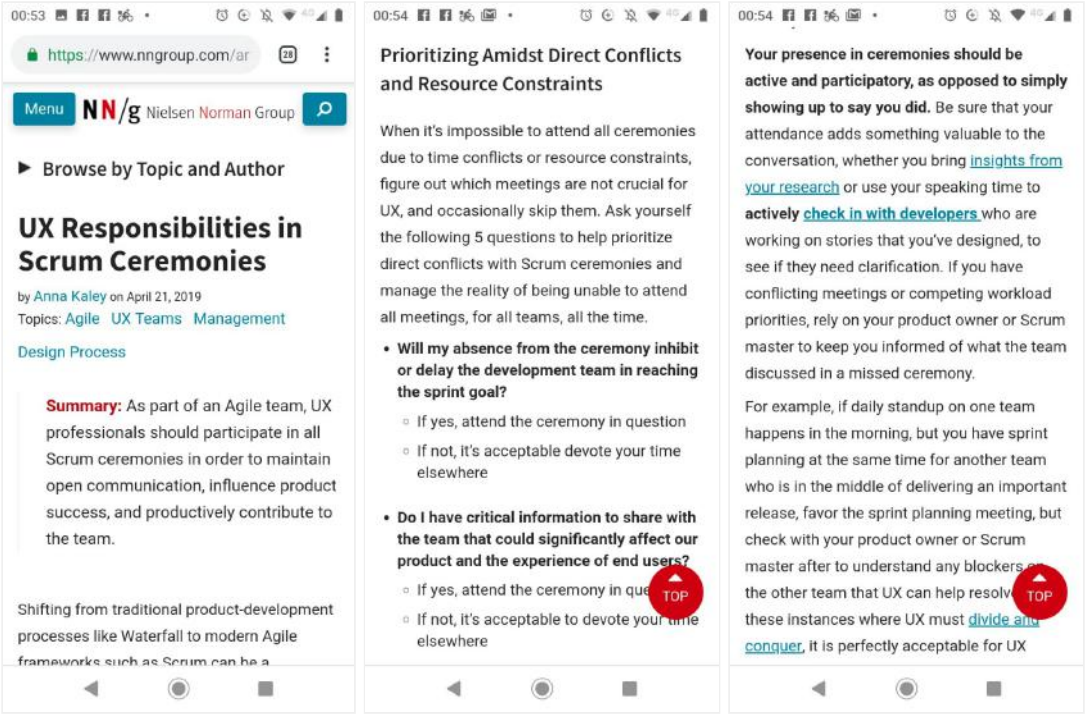
[32] Exibição de conteúdo que ocupa o *viewport* inteiro
[33] Janela que abre sobre a página e bloqueia o conteúdo "em baixo". Ela só pode ser fechada por ação do usuário

3.4.2. Nielsen Norman Group

Assim como artigos do Medium, os textos no *site* Nielsen Norman Group conseguem, com poucas variações tipográficas e diferentes formatações de texto, diferenciar e destacar conteúdos. O contraste tipográfico em questão é maior, pelo uso de pesos como bold e regular. Os links são claramente destacados pela cor azul e, quando localizados dentro da massa de texto, do sublinhado.

Já que se trata de um artigo relativamente longo, há uma apresentação do conteúdo do texto por um breve resumo introdutório. Este detalhe assim como o uso de perguntas e respostas em tópicos para tornar a comunicação mais direta, distingue em funcionalidade o *site* do Medium.

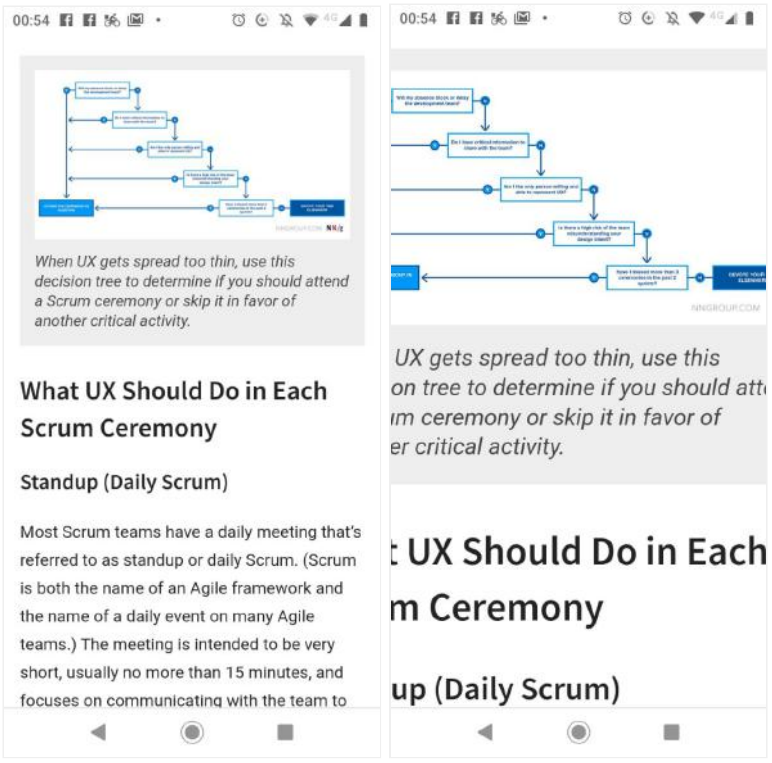
Quanto à navegação dentro do artigo, acho que o *floating button*[34] não oferece uma efetiva melhora de experiência, pois, quando no "topo", o usuário tem à sua disposição as tags dos temas abordados que levam para seções especiais e, ao final, ele possui links para outros artigos relacionados. Ainda no meio do texto, os links levam o leitor para outros artigos dentro do site. Por mais que o conteúdo seja extenso, não houve refino para navegação dentro dele.



Figuras 13, 14 e 15 - *Prints* do site Nielsen Norman Group

[34] Botão flutuante sobre a interface que acompanha a navegação

O artigo faz uso de poucas imagens, com seu estilo visual planejado para emular o conteúdo escrito (mais especificamente no caso do fluxograma a seguir), mas não houve cuidado para o consumo em *smartphones*. A imagem não abre em *full screen*, por isso, para vê-la com mais detalhes, é preciso "dar zoom" na tela, o que prejudica a visão como um todo da ilustração.

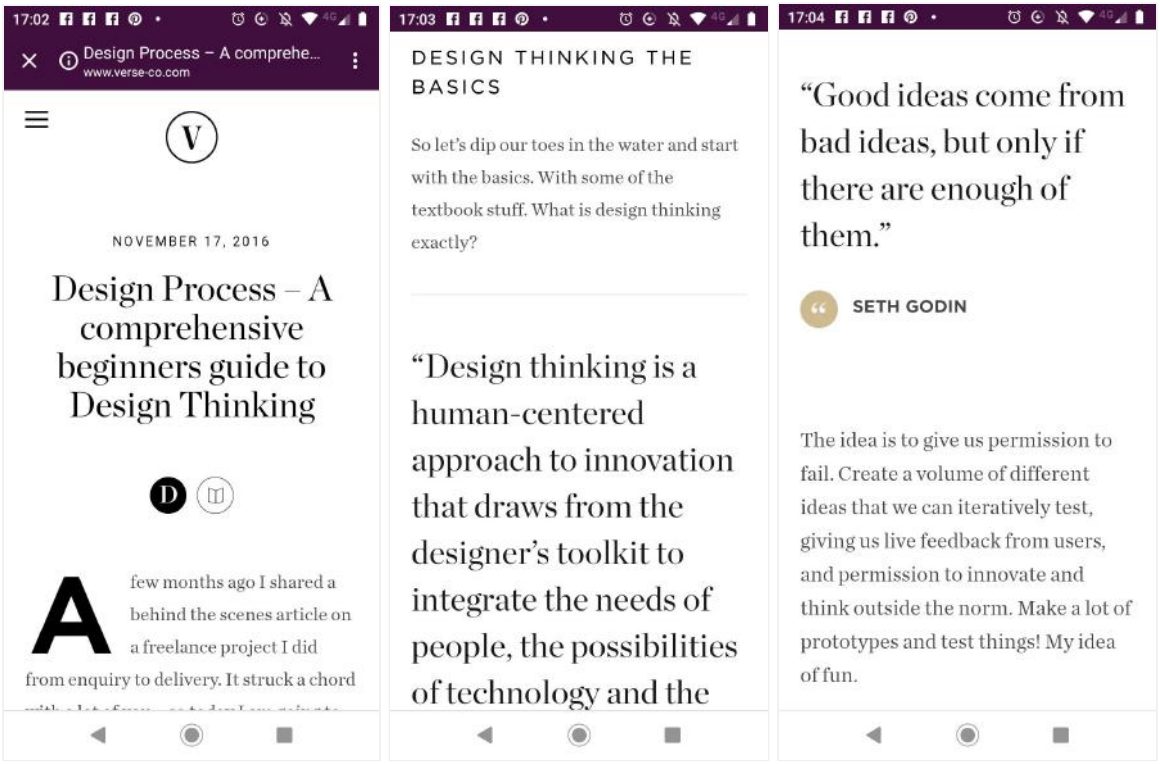


Figuras 16 e 17 - *Prints* do site Nielsen Norman Group

3.4.3. Verse

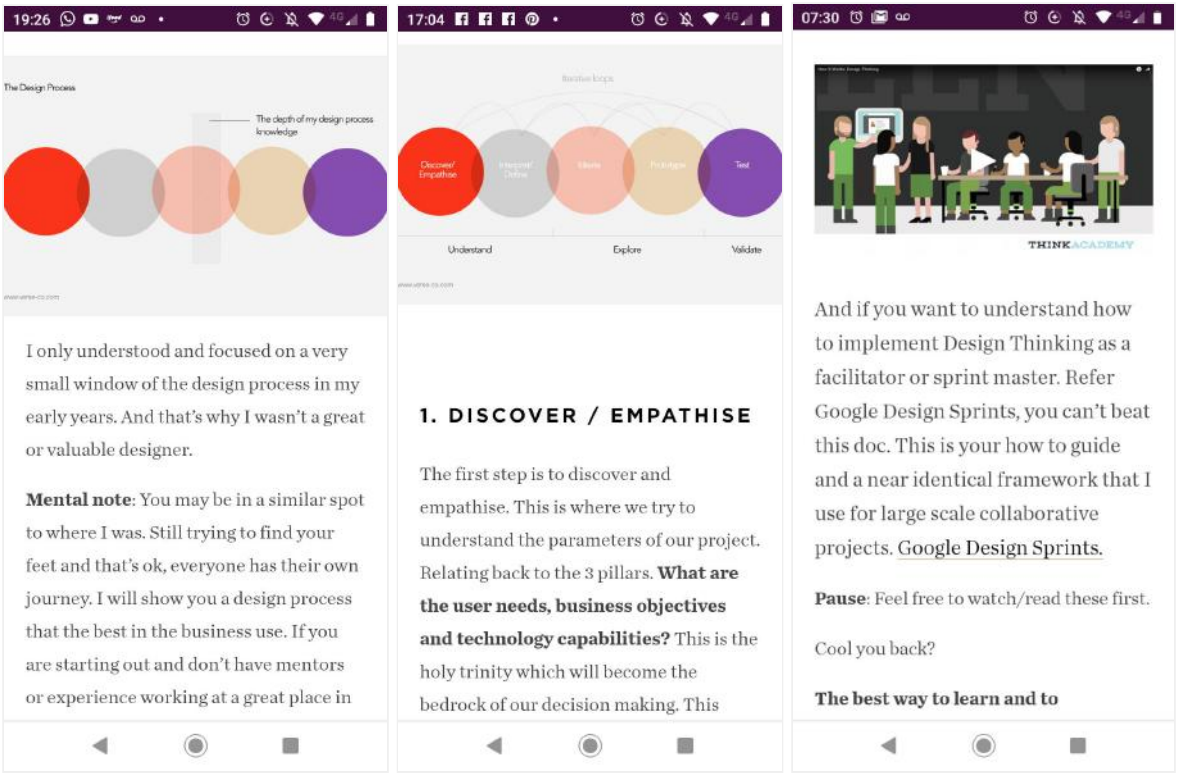
O que chama mais atenção no caso deste artigo é o seu refino estético e o uso de formas e cores simples para representação de conceitos abstratos. As escolhas tipográficas são muito interessantes pela sua elegância e pela combinação entre diferentes tipos e alinhamentos: apenas o título está centralizado enquanto o conteúdo do texto corrido é alinhado à esquerda. É interessante ainda o uso de citações em destaque como recurso para introdução de novas ideias ao texto. Mais uma vez, é possível ver o uso de diferentes pesos tipográficos para gerar contraste e destaque a certas partes do texto e os links são discretamente sublinhados para se diferenciarem dentro do texto.

A escrita informal, porém direta com verbos no imperativo, é uma forma de aproximar o conteúdo do leitor. O autor compartilha vivências pessoais assim como objetivamente enumera conteúdos e os destaca sugestões práticas. Achei interessante o uso de tópicos ao longo do texto; pontos destacados como "dicas", "nota mental" ou "pause" são formas de quebrar a leitura e chamar a atenção do leitor especialmente quando usados ao final de grandes parágrafos.



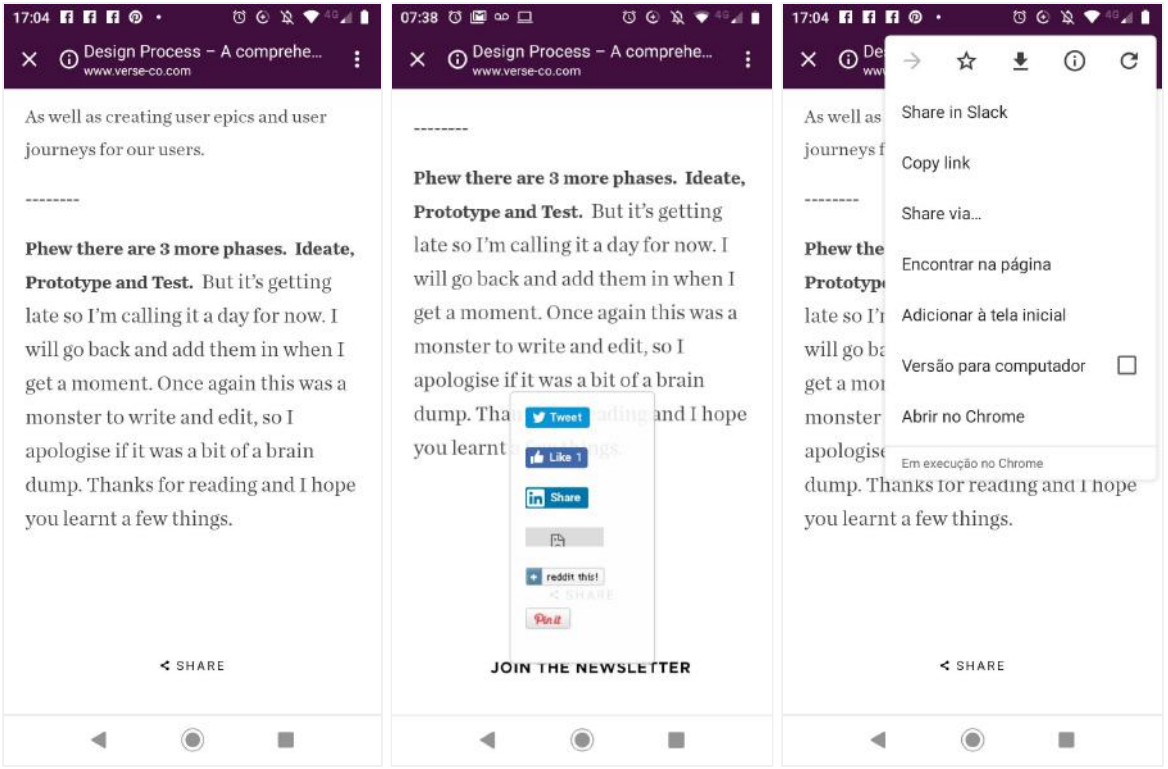
Figuras 18, 19 e 20 - *Prints* do site Verse

Quanto às ilustrações, a semelhança entre elas criou consistência visual e ajudou a criar uma certa identidade para o artigo. Os recursos visuais usados - formas geométricas simples e uma mesma paleta de cores - unificam a interface e passam a ideia de autoria - de que foi o autor que desenhou tudo isso. O único recurso que saia dessa identidade foi um vídeo embedado, mas a justificativa do autor faz sentido: Por que ele explicaria mais uma vez este conteúdo se existe um material pronto para isso? Achei interessante a opção dele por unir ilustração daquilo que é "original" do texto dele à referências de recursos já prontos. Acredito que exibir e disponibilizar referências ajude a passar credibilidade para o texto.



Figuras 21, 22 e 23 - *Prints* do site Verse

É interessante ainda reparar na disponibilização explícita de um link para compartilhamento ao final do artigo; apesar desta opção existir dentro do browser.

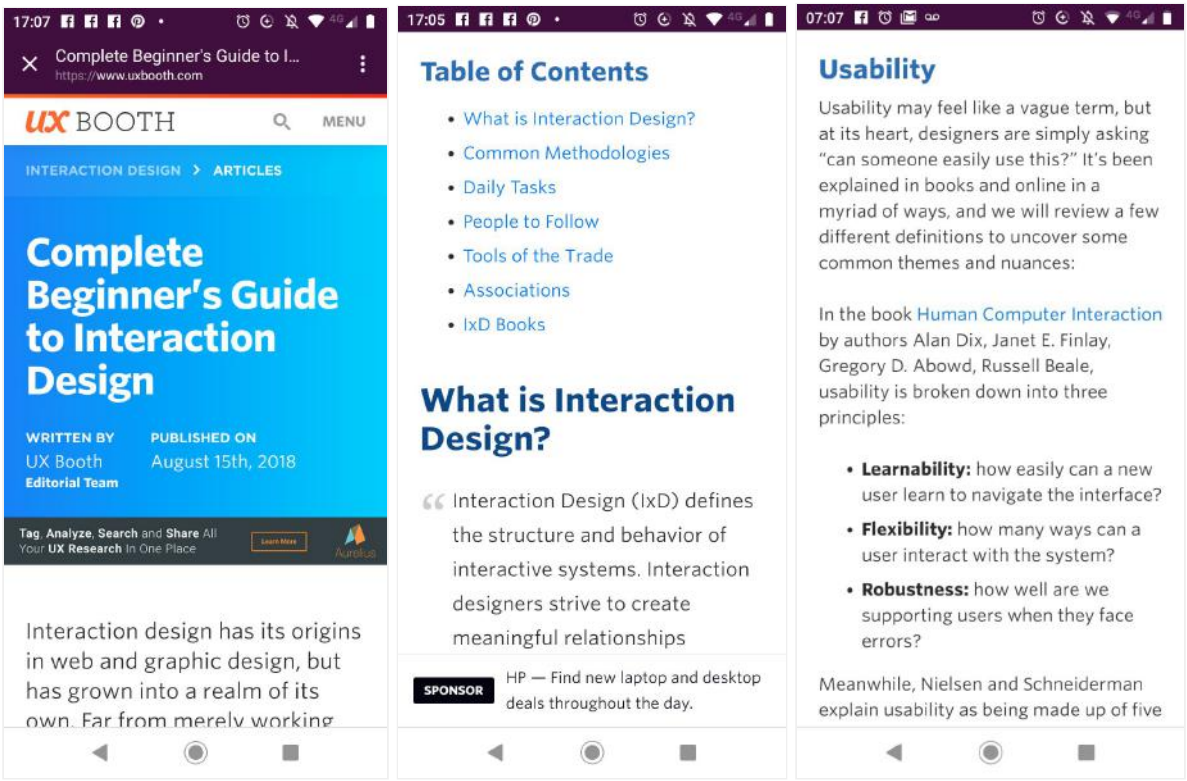


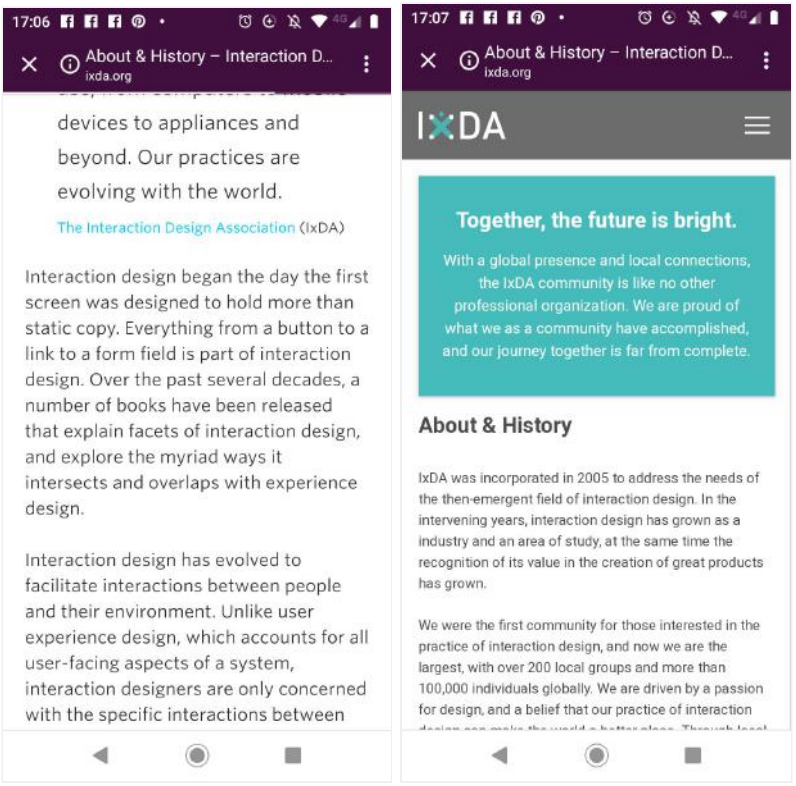
Figuras 24, 25 e 26 - *Prints* do site Verse

3.4.4. UX Booth

Entre todas as referências, esta é a que mais se adequa a um guia completo para iniciantes. Seu conteúdo é pautado na introdução de conceitos básicos e é voltado para o agrupamento de dicas e referências. Assim como os demais sites, ele usa diferentes formatações de texto como tópicos, texto corrido e olhos de texto, além de diferentes pesos tipográficos para contraste.

O site explicitamente discrimina os links pelo uso da cor azul e, apesar de ser o único que utiliza este recurso para navegação interna pelo conteúdo, não existe uma diferenciação visual entre este tipo de link e os demais que levam para sites externos. Esta distinção fica por conta dos blocos de texto onde estes links se encontram: Os links dentro de "Table of Contents" são para navegação por partes do artigo, porém eles só estão disponíveis no topo, antes de qualquer leitura; já os links dentro do texto, levam para sites externos onde o leitor pode ter acesso à referência.





Figuras 27, 28, 29, 30 e 31 - *Prints* do site UX Booth

3.4.5. Conclusão

Para sites que tratam de conteúdos tão específicos e conceitos abstratos, não foram encontrados muitos exemplos de diagramas ou textos que oferecessem uma explicação introdutória. Basicamente, estes artigos tendem a ser voltados para um público que já tem algum conhecimento e usam *links* para indicar leituras complementares, onde mais termos técnicos e específicos são apresentados ao leitor, uma vezes, sem explicação prévia. Além disso, o uso de imagens não é direcionado a *smartphones*, apesar do consumo via *mobile* ser muito popular.

3.5. Escopo

Como escopo do site, foram definidas: Informações de boas práticas comportamentais e técnicas para se trabalhar dentro de métodos ágeis; introdução a alguns conceitos básicos de UX e diagramas que tangibilizassem conteúdos abstratos. Estas interfaces são construídas sob o princípio de *mobile-first*[35].

[35] Prática associada à construção de interfaces direcionadas para consumo em *smartphones*

3.6. Construção do site

3.6.1. Naming

Ao retornar ao material das entrevistas para realizar as análises, todo o conteúdo reunido como "boas práticas comportamentais" parecia ser resumido pela expressão "atitude ágil" dita em uma das entrevistas.

O termo resume bem o que se precisa fazer para trabalhar em métodos ágeis. No sentido literal, "atitude" significa "Maneira de se comportar, agir ou reagir, motivada por uma disposição interna ou por uma circunstância determinada; comportamento" (DICIONÁRIO ONLINE DE PORTUGUÊS, [21--]).
No sentido figurado, significa ter autonomia, proatividade e intenção em suas ações. Atitude é um termo que remete a atividades práticas e que "(...) se diferencia [do termo] postura pelo maior grau de concretude dos objetos a que se refere (...)" (WIKIPÉDIA, 2019). O termo "ágil" funciona como um substantivo, uma vez que, neste contexto, faz referência aos métodos ágeis, porém, esta associação é extremamente contextual e pode ser desconhecida para muitas pessoas, até porque "ágil" é um termo comumente associado à ideia de "velocidade" e "rapidez".

Para evitar más interpretações sobre o que significa "Atitude ágil", o subtítulo situa o leitor sobre o que ele estava prestes a ler. O nome final para o site foi estabelecido como: "Atitude ágil: Como designers UX podem contribuir dentro de métodos ágeis". Através deste nome já é possível transmitir do que se trata o site assim como explicitar seu objetivo.

3.6.2. Identidade Visual

A escolha pela representação visual por meio de formas geométricas simples e cores foi principalmente baseada em princípios de acessibilidade. De acordo com dados disponibilizados em 2017 pela Organização Mundial da Saúde, estima-se que cerca de 217 milhões de pessoas vivem com alguma forma de deficiência visual moderada a grave (REYNA, 2018), dentre elas, o Daltonismo que afeta cerca de 1 em 12 homens e 1 em 200 mulheres no mundo (COLOUR BLIND AWARENESS, [21--]). Por ser tão comum, a seleção das cores para este projeto tem foco em Daltonismo:

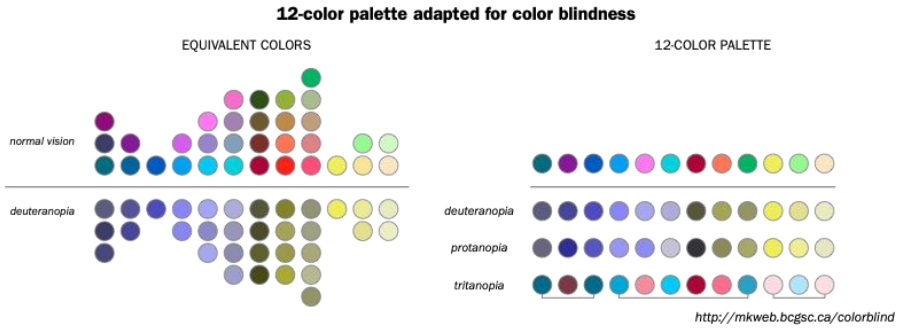


Figura 32 - 12-color palette adapted for color blindness (KRZYWINSKI, [21--])

As cores ainda possuem o papel de representar cada um dos membros dentro de ambientes ágeis:

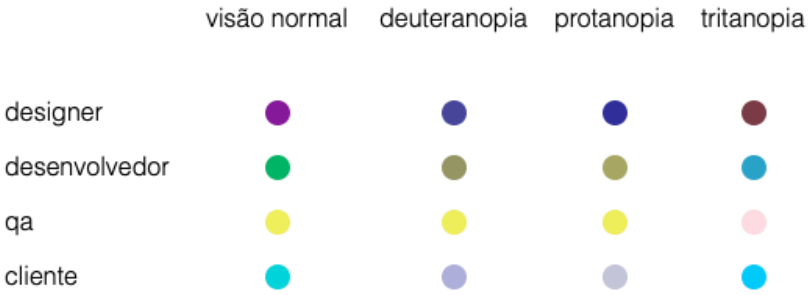


Figura 33 - Paleta de cor da interface

3.6.3. Grid e espaçamentos

A fim de acomodar o conteúdo para consumo *mobile*[36][37], caracterizado pelo seu *viewport*[38] menor, o grid foi construído considerando a menor largura de aparelho, o iPhone com dimensões 320x480px, até a maior, o monitor Chromebox 30 com dimensões 2560x1600px, e entre esses extremos, demais aparelhos *smartphones*, *tablets* e monitores. Estas escolhas tiveram como base a listagem de *viewports* feita pelo Material Design, um sistema de diretrizes para construção de interfaces feito pelo Google.

[36] Traduzido como dispositivo celular portátil
[37] Este tipo de prática é chamada *mobile-first*, onde se tem o formato *mobile* como prioridade para construção de interfaces, justamente por ser o formato mais restritivo
[38] Traduzido como janela de exibição, é o espaço correspondente à tela em aparelhos onde recursos visuais são exibidos

Com os extremos de *viewport* selecionados e a partir da consulta de boas práticas para construção de Grids (PRODUCT + PEOPLE, 2014), o grid foi construído de forma flexível e, por isso, seu maior número de colunas foi definido em 12. A escolha por este número facilita a distribuição dos elementos na interface quando ocorre a quebra do número de colunas, uma vez que ele é divisível por 2, 3, 4 e 6. Após estudos, foi definido o número de 4 colunas para o menor *viewport*.

Como resultado final, para abranger a grande variedade de tamanhos de tela e proporcionar uma boa experiência de interação em cada uma delas, a variação do grid foi definida em 4 *breakpoints*[39] onde existem 3 variações no número de colunagem e as colunas são ajustáveis.

Figura 34 - O alcance do primeiro *breakpoint* vai das larguras 320px a 639px com 4 colunas, calhas de 16px e margem de 24px; o segundo, se estende de 640px a 959px com 8 colunas, calhas de 24px e margem de 24px; o terceiro, abrange *viewports* de 960px a 1280px com 12 colunas, calhas de 24px e margem de 24px; e o quarto, com 12 colunas, calhas de 24px e margem em auto.

Junto à definição do grid, visando estabelecer um padrão dos espaçamentos entre os elementos do site, foram utilizados múltiplos de 8px, com exceção do espaçamento menor que é de 4px, tanto por uma questão de padronização do *layout* quanto por uma questão técnica. Logo, os espaçamentos usados são 4px, 8px, 16px, 24px, 32px, 40px, 48px, 56px e 64px.

No que toca à padronização, a escolha por um sistema de espaçamentos foi feita para que o número de variações de espaços seguisse uma lógica consistente, o que é importante para implementação por minimizar a possibilidade de escolhas arbitrárias que ferem o sistema estabelecido.

[39] Ponto de adaptação da interface para outro número de colunas. Ele acontece a partir da quebra da interface em um *viewport* maior ou menor.

Quanto à questão técnica, de maneira breve, a escolha foi feita, pois a estrutura de CSS[40] de uma página HTML[41] pode ser feita usando a unidade de construção rem, cujo tamanho base é 16px (EXPERIMENTS..., [21--]). A limitação dos espaçamentos em múltiplos de 8, com exceção da menor unidade ser 4px, mantém certo grau de consistência com a unidade base de rem por 16 ser múltiplo de ambos os números, o que facilita a implementação.

Figura 35 - Sistema de espaçamentos.

3.6.4. Família tipográfica

A escolha tipográfica foi pautada em aspectos técnicos para performance *web* e em parâmetros de acessibilidade. Dentre as fontes de sistema[42] mais comuns, foram identificadas aquelas que melhor se adequam às diretrizes de acessibilidade.

A fonte titular é Helvetica, uma fonte que apesar de não ser comum a todos os navegadores, já é presente na maioria deles, e que tem como opção de *fallback*[43], em ordem de preferência,[44] para Arial, Verdana, Tahoma e demais fontes *sans serif*.

Figura 36 - Família tipográfica

[40] CSS é a sigla para Cascading Style Sheets. É usado para adicionar estilo a um documento web.
[41] HTML é a sigla para Hypertext Markup Language. É uma linguagem para desenvolver websites.
[42] São fontes nativas do browser
[43] Termo usado para designar o planejamento com os procedimentos necessários para restaurar um sistema de volta a condição operacional
[44] Cito esta especificação aqui, porque ela seria necessária para se fazer o site real

Fontes *sans serif* são boas opções para acessibilidade, pois seus desenhos são simples e básicos, sem decorações que podem atrapalhar a leitura, assim como geralmente performam melhor em aparelhos eletrônicos e em telas com baixa resolução (BUREAU OF INTERNET ACCESSIBILITY, 2017). Ainda considerando aspectos de acessibilidade, o menor tamanho de fonte foi definido em 16 px e, ao longo do texto, são usados diferentes pesos tipográficos para destaques.

3.6.5. Família iconográfica e componentes

Os ícones usados são do Material Design, uma vez que estes são amplamente usados e conhecidos por meio de sites projetados pelo Google e pelo sistema operacional Android, o mais popular no Brasil (DEVICE ATLAS, 2019).

Figura 37 - Lista de ícones usados na interface

De mesma forma, os componentes usados para a construção das telas seguem as diretrizes escritas pelo Material Design respeitando suas normas de acessibilidade e usabilidade.

3.6.6. Linguagem

Para construção do texto, a escolha foi por uma linguagem direta e amigável e pela distribuição do conteúdo entre parágrafos e tópicos. Os parágrafos são para construir a narrativa entre os temas, ambientando e introduzindo o leitor, enquanto os tópicos são usados para expor dicas de maneira pontual.

Por ser um conteúdo específico, manter termos especiais em inglês foi a escolha, uma vez que a pesquisa mostrou que esta é a forma coloquial utilizada pelos profissionais da área. Este conteúdo é escrito em diferentes tamanhos e pesos tipográficos para contraste.

3.6.7. Navegação

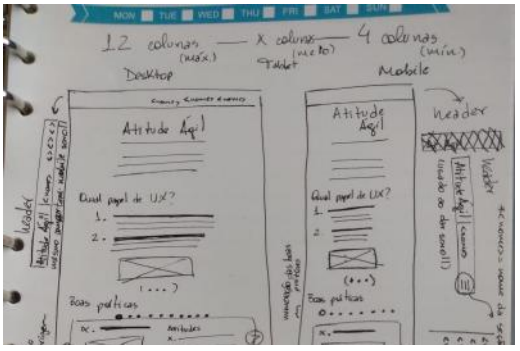
O conteúdo do site foi determinante para o desenho de sua navegação. Como se trata de um conteúdo explicativo, a ordem em que o texto se desenvolve é importante para a compreensão e construção dos conceitos ali citados. Por isso, sua navegação é simples e acontece principalmente por meio de *scroll*[45], porque esta orientação emula a apresentação sequencial do conteúdo.

Considerando a grande quantidade de informações, o *top bar*[46] acompanha a navegação por *scroll* e exibe o nome da seção em que o usuário se encontra para contextualizá-lo. Nele, ainda tem um *menu burger*[47] que exibe uma *sidebar*[48] com os títulos *linkados* para cada uma das seções do site, facilitando a navegação dos usuários sobre o conteúdo.

Alguns grupos de informação estão associados a ilustrações. Estas podem ser exibidas em *full screen* e abrem como modais sobre o site. Pensando na limitação de exibição do conteúdo em *viewports mobile* e para não deixar a experiência exaustiva pela grande repetição de *scrolls*, alguns conteúdos são exibidos em cards e exigem *swipe*[49] para serem exibidos.

3.6.8. Wireframes

Com o conteúdo definido, rascunhos foram feitos de de modo a distribuí-lo pela interface. Ele foi disposto para que coubesse dentro do *viewport* do usuário, uma vez que a relação entre imagem e texto é muito presente.



Figuras 38 e 39 - Rascunho das interfaces

[45] Interação em *smartphones* caracterizada por "deslizar" verticalmente o conteúdo no *viewport*
[46] Barra superior que fornece conteúdo e ações relacionadas à tela atual. Ela é usado para *branding*, títulos de tela, navegação e ações.
[47] Ícone que representa menu
[48] Barra lateral que pode ser fixa no *viewport* ou exibida quando aciona
[49] Gesto interativo comum em *smartphones* caracterizado por "deslizar" horizontalmente o conteúdo dentro do *viewport*

3.6.9. Produto final

A organização do conteúdo do site respeita a seguinte ordem de apresentação:

- **Introdução**, onde é explicado de maneira breve o que é UX e o objetivo do site;
- **Agilidade**, onde é explicado o que é o Manifesto Ágil e o que é Scrum junto de dicas comportamentais frente às especificidades das cerimônias Scrum e
- **Boas práticas de Design**, cujos subtópicos são "Pensando Design", onde são apresentadas dicas relacionadas ao *Design Thinking*, e "Fazendo Design", onde estão dicas técnicas que abrangem o foi chamado de "racional" do design, componentes e documentação.

A tela final do site e seus componentes podem ser encontrados neste link: https://drive.google.com/open?id=1mcaH9vb9Lv3RsvM5nf3qNi_9NWN_Ctwu

Esta porção da monografia está limitada à exposição do artefato visual.

Figuras 40, 41 e 42 - Na primeira figura, exposição do título e ilustração com seções do site; na segunda e terceira imagens, ilustração da navegação no momento quando o top bar de navegação com menu fica disponível na tela.

Figuras 43 e 44 - Imagens ilustrativas dos tópicos e subtópicos do site. A primeira exibe os *top bars* e a segunda o menu com os links das sessões.

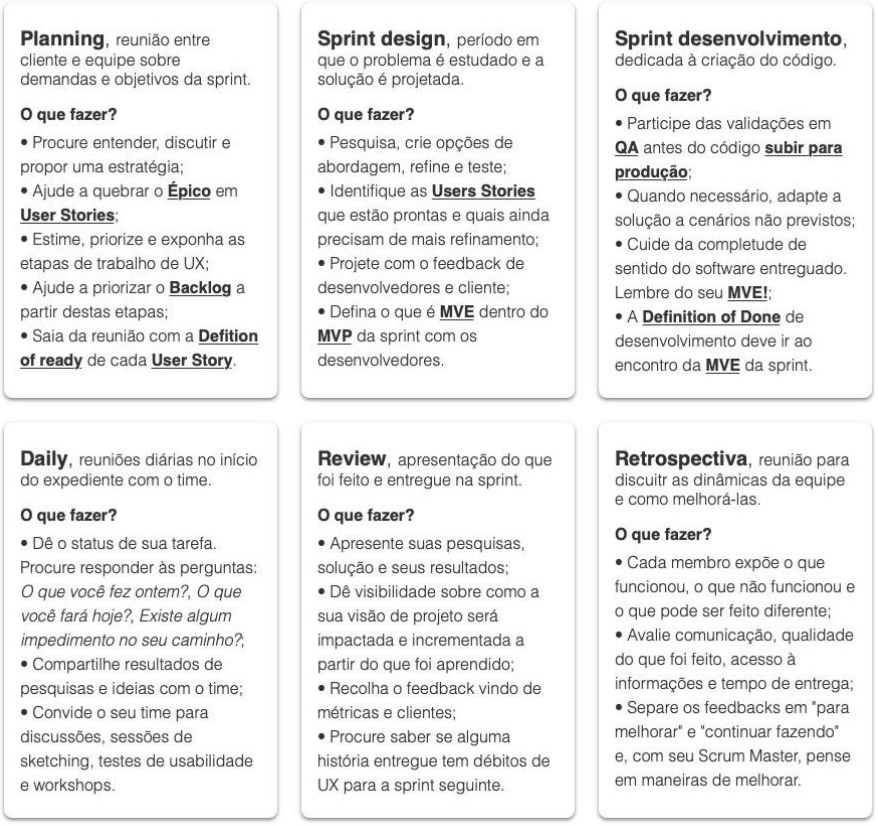
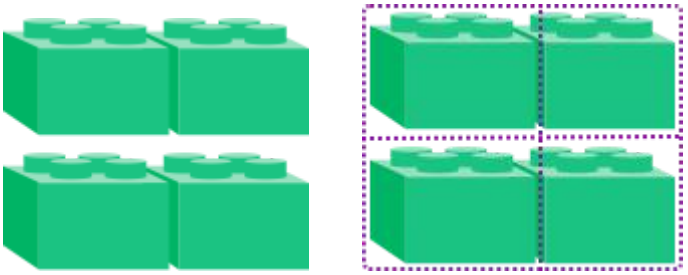


Figura 45 - Cards para *swipe* contendo informações sobre as cerimônias Scrum



Figuras 46 e 47 - Ilustrações desenvolvidas para explicar a construção de software.

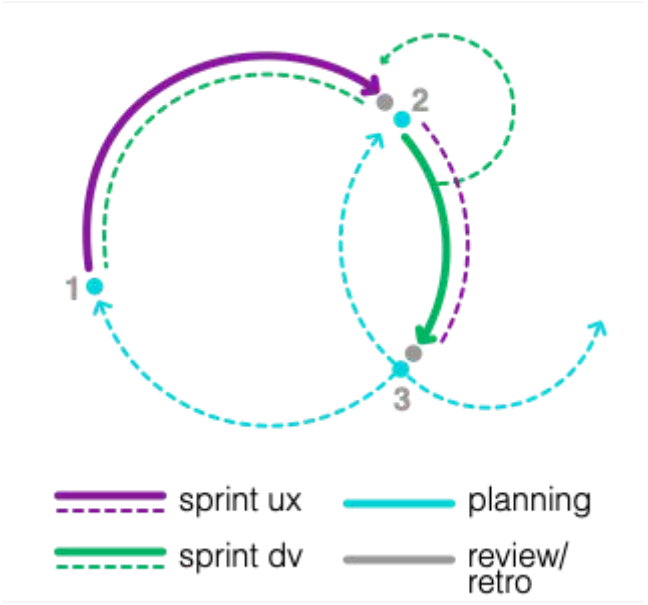
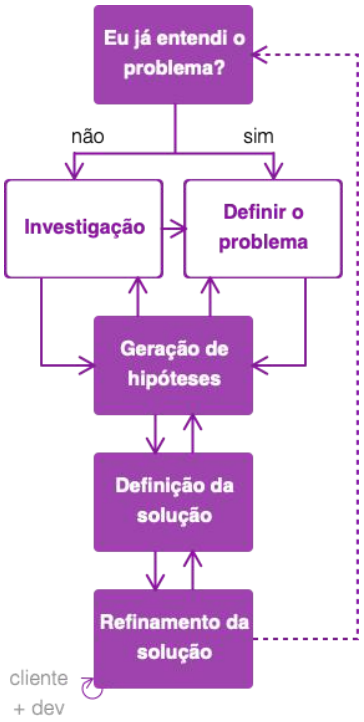
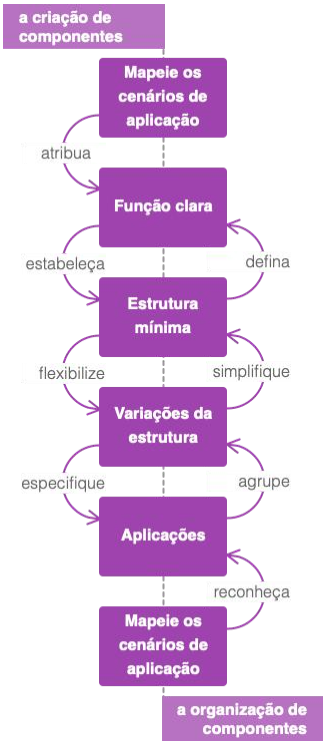


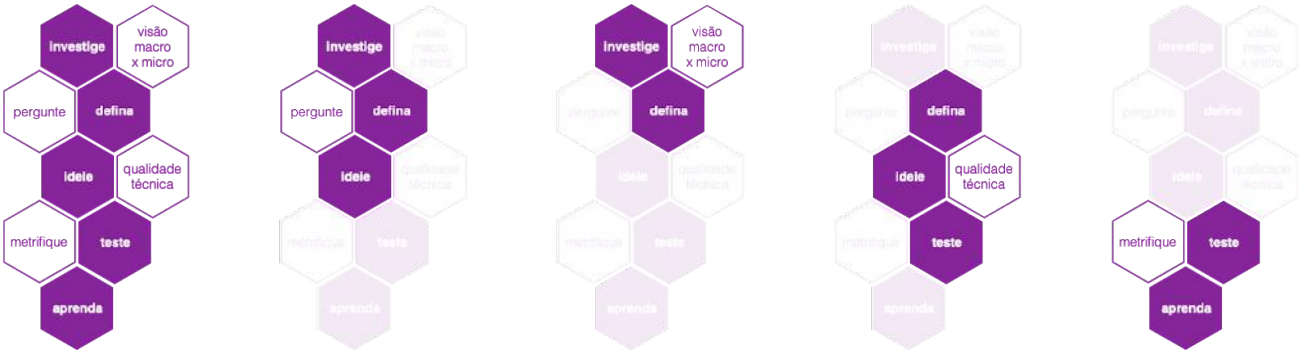
Figura 48 - Diagrama elaborado para "localizar" a sprint de design em meio ao desenvolvimento.

Figura 49 - Adaptação do diagrama de Venn para ilustrar a interseção entre as áreas de negócio, desenvolvimento e UX.

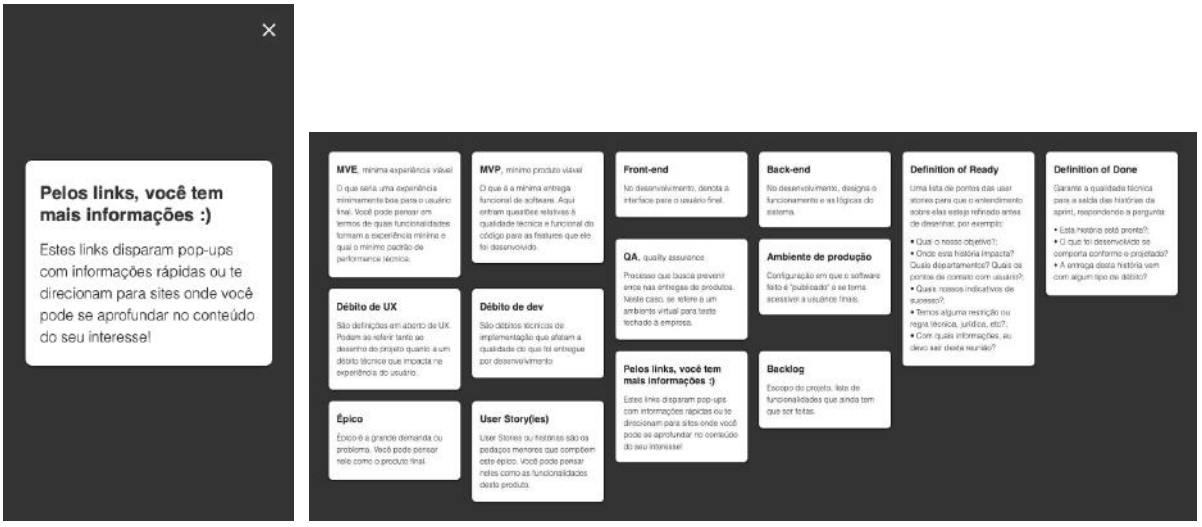




Figuras 50, 51 e 52 - Diagramas criados para explicar respectivamente: etapas clássicas de Design Thinking, modularização de componentes e os componentes da equipe Scrum.



Figuras 53 - Diagrama que relaciona etapas de Design Thinking somadas à algumas boas práticas de design UX e seu desenvolvimento em animação para o site.



Figuras 54 e 55 - Aplicação de pop-up e listagem de todos

4. CONSIDERAÇÕES FINAIS

O aprendizado durante este projeto foi imenso. Foi dada a oportunidade de aprofundamento em um tema tão atual e relevante, assim como a possibilidade por explorar maneiras de tangibilizar e transmitir conceitos abstratos. Foi extremamente interessante desenvolver um estudo mais teórico e pragmático sobre como se fazer design e explorar outras de suas vertentes para além da visual, como pesquisa, curadoria, escrita e definição de processos. Estas últimas são partes integrantes do produto final e mesmo, no caso deste trabalho, predecessoras ao artefato visual.

Foi um projeto que levou à redefinição pessoal do papel do designer para a autora, agora, que ela se reconhece mais como parte de um sistema em que os limites entre as funções e responsabilidades de cada um de seus agentes parecem cada vez mais tênues e substituídas por uma abordagem mais colaborativa.

Como planejamentos futuros, há o desejo de testar a eficiência comunicativa destas interfaces e ilustrações com designers e desenvolvedores seniores e juniores, de modo que ajustes e adaptações possam ser feitos.

Referências

AMMOURI, Yahya Mohammed. **Agile Software Development Basics and fundamentals**. [*S. l.*], 15 dez. 2015. Disponível em: https://www.codeproject.com/Articles/1064114/Agile-Software-Development-Basics. Acesso em: 2 nov. 2018.

ARMITAGE, John. **Are agile methods good for design?**. [S.l.: s.n.], 2004. 18 p. Disponível em: <https://www.researchgate.net/publication/220382925>. Acesso em: 20 dez. 2018.

BECK, Kent; BEEDLE, Mike et al. **Manifesto para Desenvolvimento Ágil de Software**. 2001. Disponível em: <http://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em: 21 nov. 2018.

BUCHANAN, Richard. **Wicked Problems in Design Thinking**. [*S. l.*], 1992. Disponível em: http://web.mit.edu/jrankin/www/engin_as_lib_art/Design_thinking.pdf. Acesso em: 2 fev. 2019.

BUREAU OF INTERNET ACCESSIBILITY. **BEST FONTS TO USE FOR WEBSITE ACCESSIBILITY**. [*S. l.*], 20 maio 2017. Disponível em: https://www.boia.org/blog/best-fonts-to-use-for-website-accessibility. Acesso em: 3 jun. 2019.

CHEGG STUDY. **Question 2) Workload & Stress Vs. Performance**. [*S. l.*], [21--]. Disponível em: https://www.chegg.com/homework-help/questions-and-answers/question-2-workload-stress-vs-performance-discussion-essay-50-pts-figure-1-yerkes-dodson-h-q27129601. Acesso em: 2 nov. 2018.

COLLABNET. **A Glossary of Scrum / Agile Terms**. South San Francisco, California: [*s. n.*], [21-?]. Disponível em: https://www.collab.net/sites/all/themes/collabnet/_media/pdf/gl/CollabNet_glossary_scrum_agile_terms.pdf. Acesso em: 23 fev. 2019.

COLOUR BLIND AWARENESS. **Colour Blindness**. [*S. l.*], [21--]. Disponível em: http://www.colourblindawareness.org/colour-blindness/. Acesso em: 20 jun. 2019.

CPRIME. **Agile Glossary**. [*S. l.*: *s. n.*], 2013. Disponível em: https://www.cprime.com/wp-content/uploads/2013/10/cPrime-Agile-Glossary.pdf. Acesso em: 23 fev. 2019.

DEVICE ATLAS. **Android v iOS market share 2019**. [*S. l.*], 31 maio 2019. Disponível em: https://deviceatlas.com/blog/android-v-ios-market-share. Acesso em: 9 jun. 2019.

FORBES. 8 Common Causes Of Workplace Demotivation. [*S. l.*], 20 jan. 2014. Disponível em: https://www.forbes.com/sites/work-in-progress/2014/01/20/8-common-causes-of-workplace-demotivation/#4636cac842c6. Acesso em: 10 jan. 2019.

GARRETT, Jesse. **The Elements of User Experience**: User-Centered Design for the Web and Beyond. 2. ed. Estados Unidos da América: New Riders, 2011. 191 p.

GOOGLE. **Material Design**. [*S. l.*], 2014. Disponível em: https://material.io/design/introduction/#. Acesso em: 8 maio 2019.

HEFLO. Estrutura de trabalho (Framework). [*S. l.*], [21-?]. Disponível em: https://www.heflo.com/pt-br/definicoes/estrutura-de-trabalho/. Acesso em: 27 fev. 2019.

HIGHSMITH, Jim. History: The Agile Manifesto. In: BECK, Kent; BEEDLE, Mike et al. **Manifesto para Desenvolvimento Ágil de Software**. 2001. Disponível em: <http://agilemanifesto.org/iso/ptbr/manifesto.html>. Acesso em: 21 nov. 2018.

IDEOU. **Design Thinking**. [*S. l.*], [21-?]. Disponível em: https://www.ideou.com/pages/design-thinking. Acesso em: 7 mar. 2019.

KALEY, Anna. **UX Responsibilities in Scrum Ceremonies**. [*S. l.*], 21 abr. 2019. Disponível em: https://www.nngroup.com/articles/ux-scrum/. Acesso em: 21 abr. 2019.

KRZYWINSKI, Martin. **COLOR PALETTES FOR COLOR BLINDNESS**. [*S. l.*], [21--]. Disponível em: http://mkweb.bcgsc.ca/colorblind/. Acesso em: 3 jun. 2019.

LEMAY, Matt. **Agile Is Ruining Your Product**. [*S. l.*], janeiro [21--]. Disponível em: https://medium.com/on-human-centric-systems/agile-is-ruining-your-product-3674c8afea76. Acesso em: 25 mar. 2019.

MORVILLE, Peter; ROSENFELD, Louis. **Information Architecture for the World Wide Web**. 3. ed. Estados Unidos da América: O'Reilly Media, Inc., 2006. 528 p.

MOURA, Mônica. **Design e/é Arte**. [*S. l.*], [21--]. Disponível em: https://docplayer.com.br/5494202-Design-arte-e-tecnologia-1-monica-moura-i-design-e-e-arte.html. Acesso em: 20 jun. 2019.

MUNARI, Bruno. **Design e Comunicação Visual**. 4. ed. São Paulo: Martins Fontes, 2011. 350 p.

NASH, Brooklin. **What is the Difference Between Agile vs Waterfall?**. [*S. l.*], 12 jun. 2019. Disponível em: https://www.trustradius.com/buyer-blog/difference-between-agile-vs-waterfall. Acesso em: 12 jun. 2019.

NIELSEN, Jakob. Foreword. In: MORVILLE, Peter; ROSENFELD, Louis (Org.). **Information Architecture for the World Wide Web**. 3. ed. Estados Unidos da América: O'Reilly Media, Inc.,, 2006. p. xi-xii.

LORANGER, Hoa; LAUBHEIMER , Page. **The State of UX Agile Development**. [*S. l.*], 21 abr. 2019. Disponível em: https://www.nngroup.com/articles/state-ux-agile-development/. Acesso em: 21 abr. 2019.

PROJECT BUILDER. **Guia da gestão de projetos: metodologia Waterfall**. Rio de Janeiro, 28 jun. 2017. Disponível em: https://www.projectbuilder.com.br/blog/guia-da-gestao-de-projetos-metodologia-waterfall/. Acesso em: 26 fev. 2019.

PRODUCT + PEOPLE. **Um guia completo sobre grids para design responsivo**. [*S. l.*], 15 out. 2014. Disponível em: https://brasil.uxdesign.cc/um-guia-completo-sobre-grids-para-design-responsivo-6b192fea0124. Acesso em: 8 maio 2019.

REYNA, Justin Rey. **Here's What You Need to Know About Color Accessibility in Product Design**. [*S. l.*], 11 out. 2018. Disponível em: https://uxplanet.org/heres-what-you-need-to-know-about-color-accessibility-in-product-design-aecbd0c30628. Acesso em: 3 jun. 2019.

RUBIN, Eran; RUBIN, Hillel. **Supporting agile software development through active documentation**. [S.l.]: Springer-Verlag London Limited, 2010. 117 p. Disponível em: <http://proxy.ufrj.br/>. Acesso em: 24 out. 2018.

SCHAEFFER, Chuck. **Agile versus Waterfall for CRM Implementation Success**. [*S. l.*], [21--]. Disponível em: http://www.crmsearch.com/agile-versus-waterfall-crm.php. Acesso em: 5 nov. 2018.

SELIC, Bran. **Agile Documentation, Anyone?**. 6. ed. [S.l.]: IEEE, 2009. 11 p. Disponível em: <https://ieeexplore.ieee.org/document/5287001>. Acesso em: 21 nov. 2018.

SMARTSHEET. Waterfall. [*S. l.*], [21-?]. Disponível em: https://www.smartsheet.com/waterfall-methodology. Acesso em: 24 fev. 2019.

SUTHERLAND, Jeff. **SCRUM**: A arte de fazer o dobro do trabalho na metade do tempo. Tradução: Natalie Gerhardt. 1 . ed. São Paulo: Grupo LeYa, 2014. 158 p.

TAYMOR, Emerson. **Agile Handbook.** [*S. l.*], [21---]. Disponível em: http://agilehandbook.com/agile-handbook.pdf. Acesso em: 27 dez. 2018.

THE TRIANGLE ADMIN. **Types of interviews for data collection**. [*S. l.*], 18 fev. 2016. Disponível em:

https://researcholic.wordpress.com/2016/02/18/types_of_interviews/. Acesso em: 3 mar. 2019.

WATERFALL model. [*S. l.*], [21-?]. Disponível em: https://en.wikipedia.org/wiki/Waterfall_model. Acesso em: 23 fev. 2019.

WIKIPÉDIA. **Atitude**. [*S. l.*], 8 abr. 2019. Disponível em: https://pt.wikipedia.org/wiki/Atitude. Acesso em: 8 maio 2019.

ZABAN, Yuri. **Design Thinking: Método Duplo Diamante**. [*S. l.*], 25 dez. 2018. Disponível em: https://webframe.com.br/design-thinking-metodo-duplo-diamante/. Acesso em: 16 jun. 2019.

Glossário

Agile: Referente à Metodologia ágil
Back end: Porção do desenvolvimento voltada para desenvolvimento das aplicações web, também chamado de back.
Backlog: Escopo de projeto, lista de features que serão desenvolvidas. Como metodologias ágeis trabalham com uma constante revisão do que é feito no software funcional, é uma lista que pode sofrer alterações ao final de cada sprint, mas não durante ela para manter o time focado e on track do que acontece.
Benchmark: Processo de busca pelas melhores práticas dentro de uma indústria; pesquisa comparativa entre competidores
Bottom app bar: Barra inferior de aplicativos que fornecem acesso a links de navegação. Idealmente, esta barra é composta minimamente por 3 links e, no máximo, por 5.
Branding: Gestão da marca de uma empresa; criação e manutenção da personalidade, visual e aplicação da marca
Bugs: Defeitos de software
Core: Núcleo, cerne, âmago
Daily: Traduzido literalmente como "diário". Na metodologia Scrum, refere-se a reuniões de time realizadas diariamente
Daily standup: Na metodologia Scrum, refere-se às reuniões diárias do time nas quais seus participantes ficam de pé. Este método parte da premissa de que o desconforto as deixa mais eficientes e objetivas.
Demos: Encurtamento de demonstração, neste contexto, amostra de software
Dev: Nome usual para desenvolvedor
Doing: Traduzido como "fazendo", "em processo". No contexto deste texto, o termo é usado em associação à metodologia *Kanban* e aos estados *To Do* e *Done*.
Done: Traduzido como "finalizado", "terminado". No contexto desta texto, o termo é usado em associação à metodologia *Kanban* e aos estados *To Do* e *Doing*.
Experiência *seamless*: Experiência do usuário que se dá de maneira fluida e integrada

Fallback: Termo usado para designar o planejamento com os procedimentos necessários para restaurar um sistema de volta a condição operacional
Features: Funcionalidades do software
Feedback: Considerando a relação homem-máquina, é referente à resposta, reação do sistema para o usuário
Floating button: Botão flutuante sobre a interface que acompanha a navegação
Flow: Fluir, manter-se em de forma constante e ininterrupta em um fluxo
Framework: "Estrutura de trabalho na modelagem de processos, uma estrutura de trabalho é qualquer associação planejada entre os modelos para atender uma política, desenho ou requisito de usabilidade." (HEFLO, [21-?])
Front end: Porção do desenvolvimento voltada para construção da interface com o usuário final, também chamado de front
Full screen: Exibição de conteúdo que ocupa o viewport inteiro
Gap: Lacuna
Hands on: Expressão para "mão na massa", aprender fazendo
Input: Traduzido literalmente como "entrada". De acordo com o contexto, pode ser interpretado como "a entrada/coleta de informações", por exemplo. Geralmente está atrelado à uma relação de input-output.
Insights: Conhecimento, compreensão
Kanban: Metodologia Lean de gerenciamento de projeto. É baseada na determinação de "estados" para as tarefas: *To Do* (por fazer), *Doing* (fazendo) e *Done* (feito).
Kick-off: Pontapé, ponto de partida
Menu *burger*: Ícone que representa menu
Mindset: Mentalidade, modelo mental
Mobile: Dispositivo celular portátil
Modal: Janela que abre sobre a página e bloqueia o conteúdo "em baixo". Ela só pode ser fechada por ação do usuário
Motion Design: Design de animação, de interação pela determinação de comportamentos e feedbacks de tela
Pain-points: Pontos de dificuldade
Print: No contexto, captura da imagem em tela

Prod.: Usado frequentemente na monografia junto a expressões como "colocar em prod.", "está em prod.", o termo se refere ao ambiente de desenvolvimento chamado Produção; é o ambiente "real" que tem contato com o usuário final
Product Owner: Dentro da metodologia Scrum, desempenha o papel do dono do produto, responsável por receber as demandas do cliente e passar para a equipe
Release: Lançamento para público
Review: Revisão sobre acontecimentos passados. No caso do Scrum, é o nome dado à reunião que apresenta e avalia as entregas da *sprint* que foi entregue
Scroll: Gesto interativo comum em smartphones caracterizado por "deslizar" verticalmente o conteúdo dentro do viewport
Scrum: Uma das diversas metodologias que seguem os princípios ágeis para gestão de trabalho e desenvolvimento de software
Scrum Master: Dentro da metodologia Scrum, desempenha o papel de mestre Scrum, responsável por melhorar os processos de trabalho entre os membros do time Scrum
Sidebar: Barra lateral que pode ser fixa no viewport ou exibida quando acionada
Skill: Habilidade
Smartphone: "Um telefone celular que executa muitas das funções de um computador, geralmente com uma interface touchscreen, acesso à Internet e um sistema operacional capaz de executar aplicativos baixados." (OXFORD, [200-])
Sprint: Pequenos ciclos de iteração dentro do Scrum
Stakeholder(s): Público estratégico; todos aqueles que têm interesse, atuam ou são impactados em um projeto
Stand alone: Site autossuficiente
Target: Alvo, objetivo
Top bar: Barra superior que fornece conteúdo e ações relacionadas à tela atual. Ela é usado para branding, títulos de tela, navegação e ações.
To Do: Traduzido como "por fazer". No contexto desta monografia, o termo é usado em associação à metodologia *Kanban* e aos estados *Doing* e *Done*.
Top-down: Sinônimo de para estrutura empresarial vertical, onde as decisões vem de uma esfera de gestão (top) para o nível de execução (down).

Touchpoint: Ponto de contato. Qualquer momento de interação de um consumidor com uma empresa
Update: Atualização
UX: Nome usual para designer de experiência do usuário. Termo vem do inglês *User Experience*
Viewport: Traduzido como janela de exibição, é o espaço correspondente à tela em aparelhos onde os recursos visuais são exibidos
Waterfall: Também conhecido como método em cascata. No contexto de metodologias de trabalho, é o modelo caracterizado por progressão sequencial e linear entre suas etapas
Wicked problems: Traduzido como "Problemas ruins", são " (...) uma ‘classe de problemas sociais mal-formulados, onde a informação é confusa, onde existem muitos clientes e a tomada de decisão possui conflitos de valores, e onde as ramificações em todo o sistema são completamente confusas’.” (BUCHANAN, 1992, p. 15)
Wireframes: Wireframe é o desenho básico de uma interface onde fica evidente a estrutura do produto final. É o esqueleto sobre o qual a interface será construída.
Workflow: Fluxo de trabalho

APÊNDICE A - Roteiro e pontos de decisão para diferentes seções do questionário online

Perguntas gerais (Background)

1. Qual é a função que você desempenha hoje? **esta pergunta já direciona para diferentes partes do questionário*
    a. Designer UX
    b. Desenvolvedor
    c. Trabalho tanto com UX quanto com desenvolvimento

Perguntas gerais

=== Questionário geral direcionado dependendo da resposta **(a, b ou c) da pergunta 1** ===

2. Há quanto tempo você trabalha nessa área?
   a. Menos de um ano
   b. Mais de um ano, mas menos do que 5 anos
   c. Mais de 5 anos, mas menos do que 10 anos
   d. Mais de 10 anos, mas menos do que 15 anos
   e. Mais de 15 anos, mas menos de 20 anos
   f. Mais de 20 anos

3. Onde que você trabalha hoje?
   a. Trabalho em uma empresa grande com uma equipe multidisciplinar
   b. Trabalho em uma empresa pequena com uma equipe multidisciplinar
   c. Trabalho em uma empresa grande, mas costumo trabalhar sozinho
   d. Trabalho em uma empresa pequena, mas costumo trabalhar sozinho
   e. Depende. Vivo em rotatividade, porque sou realocado para diferentes projetos em diferentes lugares/empresas/ com diferentes pessoas
   f. Trabalho remoto para uma empresa
   g. Trabalho remoto com freelas
   h. Outro

4. Como que costuma ser seu método de trabalho?
   a. Kanban (As tarefas são feitas para cumprir demandas pontuais e do momento; Um instrumento comum usado são os quadros com listas de TO DO, DOING e DONE)
   b. Scrum (Framework ágil e adaptativo; O trabalho é dividido por "sprints", períodos de 2 a 4 semanas; Desenvolvimento e design trabalham juntos para garantir uma entrega mínima de software funcional e velocidade em suas respostas)

c. Waterfall (Clara divisão entre as etapas de estratégia, design, desenvolvimento, teste, reparos e implementação; O final dessas etapas é marcado por *handovers*)
d. Não tenho um método bem definido
e. Não sei dizer
f. Outro

Perguntas específicas

=== Questionário adaptado para resposta **a da pergunta 1** ===

a. Designer UX

5. Como que o trato de informações costuma aparecer durante o seu fluxo de trabalho?

a. Com pesquisa e coleta de informações na busca por insumos para minhas decisões de design/projeto
b. Com a curadoria de informações; momento em que organizo e seleciono informações relevantes para o projeto/outras pessoas envolvidas
c. Quando tangibilizo visualmente os conceitos e a arquitetura de informação e/ou defino lógicas de funcionamento (navegação, comportamentos, etc.)
d. Particularmente, não acho que lido com a manipulação de informações… Geralmente, demandas chegam até mim e eu as executo
e. Não sei dizer
f. Outro

6. Quais materiais você costuma produzir?

a. Mapas de fluxo
b. Telas
c. Pesquisas
d. Protótipos interativos
e. Uma biblioteca com os assets
f. Outro

7. Poderia me dizer por que você recorre especificamente a essas formas de representação? *Discursiva *OPCIONAL

8. A partir desses materiais, como é sua experiência para registrar as seguintes informações? **[Coluna com Fácil, Trabalhosa, Difícil, Muito difícil, Varia, Não sei dizer e Não faço]**

a. Navegação entre páginas (ou seja, as páginas sequenciais e quais links levam para elas)
b. Comportamentos interativos (ou seja, a definição de feedbacks diante de um input)
c. Lógicas de fluxo (ou seja, restrições ou permissões de acesso a determinados conteúdos)
d. Os componentes visuais do layout (isto compreende: hexadecimal das cores, os padrões de botão, os padrões de tipografia, etc)
e. O embasamento para as decisões de design que foram tomadas (como pesquisas)
f. Registrar padrões visuais e/ou abstratos de funcionamento
g. Registrar exceções visuais e/ou abstratos de funcionamento

9. Existe mais algum tipo de informação que você sente dificuldade em transmitir visualmente? *Discursiva *OPCIONAL

10. Você poderia dizer por que acha que são informações difíceis de transmitir? *Discursiva *OPCIONAL

11. Esses materiais produzidos por você são usados por outras equipes ou departamentos? Se sim, para quem eles são destinados?

a. Desenvolvedores
b. Designers
c. Cliente
d. Não faço para outras pessoas **pula para questão 14*

e. Não sei dizer para quem eles se destinam **pula para questão 14*
f. Outro

a. Desenvolvimento, b. Outros designers, c. Clientes [Questão 11]

12. Você costuma fazer documentação?
    a. Sim
    b. Não
    c. Depende da situação

a. Sim [Questão 12]

13. Por que você costuma fazer documentação?
    a. Naturalmente, ao longo do processo de design, eu acabo fazendo documentação através dos materiais que eu produzo
    b. Acho que ajuda a organizar o projeto para mim mesmx
    c. Para tangibilizar as informações para os diferentes setores/ stakeholders
    d. Para servir como legado e histórico
    e. Para servir de referencial para outras pessoas
    f. Para estabelecer um padrão para os projetos futuros
    g. Para evitar que pessoas fiquem dependentes de mim
    h. Para evitar/argumentar em discussões futuras
    i. Para que o desenvolvimento seja fiel ao que foi projetado
    j. Porque tenho um padrão claro de documentação para seguir
    k. Porque, em um dado momento, eu perco visibilidade/controle sobre o projeto
    l. Outro

b. Não [Questão 12]

13. Por quê você não costuma fazer documentação?
    a. Porque os materiais que eu produzo ao longo do processo já são suficientes
    b. Porque o workflow desorganizado é um impeditivo
    c. Porque documentar é um processo muito oneroso
    d. Porque a documentação tende a ser desatualizada
    e. Porque a documentação tende a ser ignorada

f. Não tenho necessidade, pois tenho contato direto com as partes envolvidas
g. Existem funções que “automatizam” e tangibilizam especificidades, como o inspect
h. Tenho dificuldade em selecionar/organizar/tangibilizar informações abstratas
i. Tenho dificuldade de lidar com grande volume de informações
j. Não sei o que é relevante para as demais partes envolvidas
k. Tenho que ser rápido e me concentrar na entrega
l. Tenho dificuldade em seguir ou não existe um padrão de entrega estabelecido
m. Outro

c. Depende da situação [Questão 12]

13. Sob quais situações você faz documentação?
    a. Faço apenas quando tenho tempo
    b. Faço uma documentação complementar aos materiais que foram produzidos de acordo com a necessidade do projeto
    c. Faço quando acho necessário tangibilizar informações mais abstratas que não consigo transmitir visualmente
    d. Faço caso o desenvolvimento aconteça longe de mim
    e. Faço quando sei que passarei adiante um projeto
    f. Faço quando sei que deixarei uma equipe
    g. Faço quando posso criar um padrão de entrega
    h. Faço quando tenho que seguir um padrão de entrega estabelecido
    i. Outro

14. Quais dessas ferramentas você usa durante o seu trabalho?
    a. Sketch
    b. Adobe XD
    c. Photoshop
    d. Figma
    e. Zeplin
    f. InVision
    g. Protopie

h. Axure
i. Principle
j. Abstract
k. GitHub
l. Visual Studio
m. Outro

15. Na sua opinião, quais dentre esses problemas são os que mais atrapalham ou impedem um bom controle sobre um projeto? Selecione no máximo 5 opções, por favor

a. A dificuldade de comunicação
b. Ter visibilidade do impacto de mudanças
c. Retrabalho
d. Perda de informação durante o processo
e. Falta de padrão nas entregas
f. Cada alteração ter que ser comunicada e validada com diferentes partes
g. Falta de um histórico de projeto claro
h. Acesso fácil à informação
i. Organização e compreensão das informações
j. Incompatibilidade ou diferenças entre softwares
k. Não conseguir acompanhar as demandas
l. Outro

16. Como, na sua opinião, você acha que suas ferramentas poderiam te ajudar? Selecione no máximo 5 opções, por favor

a. Tendo uma interface e interações simples e intuitivas
b. Oferecendo funções voltadas tanto para design quanto para desenvolvimento
c. Me permitindo trabalhar colaborativamente
d. Tendo inteligência de versionamento
e. Me ajudando a organizar melhor meu projeto e suas informações
f. Possuindo funções que automatizem o trabalho
g. Se integrando com minhas outras ferramentas

h. Sendo de fácil acesso/compartilhamento
i. Me dando liberdade criativa
j. Funcionando como um guia
k. Funcionando com um centralizador para os meus trabalhos
l. Outro

Opções a, b, c [Questão 5]

17. Definições de projeto mudam mesmo durante o desenvolvimento. Uma vez o projeto está sendo codado, você acha que faz sentido atualizar os materiais de design?

a. Sim, porque é importante ter as informações do projeto atualizadas e consistentes com o que está no ar
b. Sim, afinal todas as alterações tem que ser desenhadas para serem mandadas para o desenvolvimento
c. Sim, porque outras pessoas usam a documentação como referencial
d. Não, porque a documentação mais fiel é o código que está no ar
e. Não, porque uma vez que a entrega é feita, se encerra a participação do design
f. Não, porque documentação de design tende a ser estática
g. Outro

18. Você conhece/ouviu falar no vocabulário visual de Jesse James Garrett?

a. Sim, ouvi falar, mas não uso
b. Sim, eu uso
c. Não, nunca ouvi falar

19. Gostaria de fazer algum comentário final? *Discursiva *OPCIONAL

Perguntas específicas

=== Questionário adaptado à resposta **b da pergunta 1** ===

b. Desenvolvedor

5. Quais materiais você considera parte de uma entrega mínima, isto é, que são indispensáveis para que você consiga desenvolver?

a. Mapas de fluxo
b. Telas
c. Pesquisas
d. Protótipos
e. Uma biblioteca com os assets
f. Documentação complementar explicativa
g. Outro

6. A partir desses materiais, quais informações sobre o design você precisa ter?

a. Navegação entre páginas (ou seja, as páginas sequenciais e quais links levam para elas)
b. Comportamentos interativos (ou seja, a definição de feedbacks diante de um input)
c. Lógicas de fluxo (ou seja, restrições ou permissões de acesso a determinados conteúdos)
d. Os componentes visuais do layout (isto compreende: hexadecimal das cores, os padrões de botão, os padrões de tipografia, etc)
e. O embasamento para as decisões de design que foram tomadas (como pesquisas)
f. Registro explicativo de padrões visuais e/ou abstratas de funcionamento
g. Registro explicativo de exceções visuais e/ou abstratas de funcionamento
h. Outro (pode citar mais de um)

7. Existe mais algum tipo de informação (mais abstrata ou visual) que você sente que as entregas de design não contemplam? *Discursiva *OPCIONAL

8. Você fica satisfeito com os materiais de design?

a. Sim
b. Não

c. Às vezes

a. Sim [Questão 7]

9. Por que você fica satisfeito?
   a. As informações geralmente estão completas
   b. Tenho acesso à uma documentação padronizada, clara e concisa
   c. Possuo boas ferramentas de inspect
   d. Tenho um workflow claro e definido
   e. Os meios de entregas são centralizados e padronizados
   f. Não necessito ter as mesmas ferramentas que os designers
   g. Tenho integração entre ferramentas
   h. Conceitos, condições navegacionais e layout são claros
   i. Compreendo bem as informações que são passadas
   j. Outro

b. Não [Questão 7]

9. Por que você não fica satisfeito?
   a. As informações não são completas, bem definidas ou claras
   b. A documentação não é padronizada/clara/concisa
   c. Não possuo boas ferramentas de inspect
   d. Não tenho um framework claro e definido
   e. As entregas são dispersas
   f. Geralmente preciso baixar as mesmas ferramentas que os designers
   g. Não tenho integração entre as ferramentas
   h. Depende muito do designer em questão saber fazer materiais de qualidade
   i. Materiais geralmente não contemplam informações abstratas de funcionamento/ navegação
   j. Outro

c. Às vezes [Questão 7]

9. Por que você fica satisfeito com esses materiais apenas às vezes?
   a. As informações geralmente não são completas, bem definidas ou claras

b. A documentação dificilmente é padronizada/clara/concisa
c. Não tenho framework/Dificilmente o framework é seguido
d. As entregas geralmente são dispersas
e. Geralmente preciso baixar as mesmas ferramentas que os designers
f. A qualidade dos materiais depende das condições do projeto e do designer
g. Quais materiais são entregues depende das condições do projeto e do designer
h. Materiais geralmente não contemplam informações abstratas de funcionamento/ navegação
i. Às vezes preciso de uma documentação complementar aos materiais disponibilizados
j. Outro

10. Na sua opinião, quais dentre esses problemas são os que mais atrapalham ou impedem um bom controle sobre um projeto? Selecione no máximo 5 opções, por favor

a. A dificuldade de comunicação
b. Ter visibilidade do impacto de mudanças
c. Retrabalho
d. Perda de informação durante o processo
e. Falta de padrão nas entregas
f. Cada alteração ter que ser comunicada e validada com diferentes partes
g. Falta de um histórico de projeto claro
h. Acesso fácil à informação
i. Organização e compreensão das informações
j. Incompatibilidade ou diferenças entre softwares
k. Não conseguir acompanhar as demandas
l. Outro

11. Geralmente, esses materiais são disponibilizadas para você sob quais formatos?

a. PNG
b. JPEG

c. VSG
d. PDF
e. Consulto esses materiais por um programa específico
f. Outro

12. Quais dessas ferramentas você usa para consultar os materiais de design?

a. Sketch
b. Adobe XD
c. Photoshop
d. Figma
e. Zeplin
f. InVision
g. Protopie
h. Axure
i. Principle
j. Abstract
k. GitHub
l. Visual Studio
m. Outro

13. Como, na sua opinião, você acha que suas ferramentas poderiam te ajudar? Selecione no máximo 5 opções, por favor

a. Tendo uma interface e interações simples e intuitivas
b. Oferecendo funções voltadas tanto para design quanto para desenvolvimento
c. Me permitindo trabalhar colaborativamente
d. Tendo inteligência de versionamento
e. Me ajudando a organizar melhor meu projeto e suas informações
f. Possuindo funções que automatizem o trabalho
g. Se integrando com minhas outras ferramentas
h. Sendo de fácil acesso/compartilhamento
i. Me dando liberdade criativa
j. Funcionando como um guia

k. Funcionando com um centralizador para os meus trabalhos
l. Outro

14. Quais as “linguagens” de programação são as mais importantes/ usadas por você?

a. HTML
b. Java Script
c. JAVA
d. C++S#
e. Python
f. PHP
g. MySQL
h. Outro

15. Você conhece/ouviu falar no vocabulário visual de Jesse James Garrett?

a. Sim, ouvi falar, mas não uso
b. Sim, eu uso
c. Não, nunca ouvi falar

16. Gostaria de fazer algum comentário final? Algo que você gostaria que designers soubessem e levassem em consideração no momento de fazer seus materiais? *Discursiva *OPCIONAL

Perguntas específicas

=== Questionário adaptado à resposta **c da pergunta 1** ===

c. Trabalho tanto com UX quanto com desenvolvimento

5. Como lidar com esses extremos (design e código) é vantajoso no seu dia a dia de trabalho?

a. Consigo ter uma visão ampla sobre o projeto

b. Acompanho todo o ciclo de vida do projeto: desde sua concepção até sua execução
c. Tenho conhecimento sobre todo histórico do projeto
d. Faço o design já tendo em mente o que é viável de ser feito no código
e. Por centralizar ambas as funções, geralmente todas as informações passam por mim
f. Nunca ou Dificilmente fico dependente de outra pessoa
g. Nunca ou Dificilmente outras pessoas ficam dependentes de mim
h. Sinto que posso agir com mais rapidez
i. Por entender as lógicas e padrões de design, não preciso ilustrar tudo
j. Não preciso me preocupar com passar informação
k. Conheço e domino todo o conteúdo e informações de design
l. Meu código está sempre atualizado conforme a necessidade
m. Outro

6. Como é seu método de trabalho?
   a. Primeiro passo por uma etapa de design e depois vou para o código
   b. Faço o design diretamente no código, ou seja, vou fazendo o design conforme vou codando **pula para questão 10*
   c. Outro

a. Sim [Questão 7]

7. Quais materiais você geralmente produz?
   a. Mapas de fluxo
   b. Telas
   c. Pesquisas
   d. Protótipos
   e. Uma biblioteca com os assets
   f. Outro

a. Sim [Questão 7]

8. Pode nos dizer para quem eles são destinados?

a. Desenvolvedores
b. Designers
c. Cliente
d. Para mim mesmo
e. Outro

a. Sim [Questão 7]

9. Você costuma fazer documentação?
    a. Sim
    b. Não
    c. Depende da situação

a. Sim [Questão 11]

10. Por que você costuma fazer documentação?
    a. Naturalmente, ao longo do processo de design, eu acabo fazendo documentação através dos materiais que eu produzo
    b. Acho que ajuda a organizar o projeto para mim mesmx
    c. Para tangibilizar essas informações para os diferentes setores/ stakeholders
    d. Para servir como legado e histórico
    e. Para servir de referencial para outras pessoas
    f. Para estabelecer um padrão para os projetos futuros
    g. Para evitar que pessoas fiquem dependentes de mim
    h. Para evitar/argumentar em discussões futuras
    i. Para que o desenvolvimento seja fiel ao que foi projetado
    j. Porque tenho um padrão claro de documentação para seguir
    k. Porque, em um dado momento, eu perco visibilidade/controle sobre o projeto
    l. Outro

b. Não [Questão 11]

10. Por quê você não costuma fazer documentação?
    a. Porque os materiais que eu produzo ao longo do processo já são suficientes
    b. Porque o workflow desorganizado é um impeditivo

c. Porque documentar é um processo muito oneroso
d. Porque a documentação tende a ser desatualizada
e. Porque a documentação tende a ser ignorada
f. Não tenho necessidade, pois tenho contato direto com as partes envolvidas
g. Existem funções que “automatizam” e tangibilizam especificidades, como o inspect
h. Tenho dificuldade em selecionar/organizar/tangibilizar informações abstratas
i. Tenho dificuldade de lidar com grande volume de informações
j. Não sei o que é relevante para as demais partes envolvidas
k. Tenho que ser rápido e me concentrar na entrega
l. Tenho dificuldade em seguir ou não existe um padrão de entrega estabelecido
m. Outro

c. Depende da situação [Questão 11]

10. Sob quais situações você faz documentação?
    a. Faço apenas quando tenho tempo
    b. Faço uma documentação complementar aos materiais que foram produzidos de acordo com a necessidade do projeto
    c. Faço quando acho necessário tangibilizar informações mais abstratas que não consigo transmitir visualmente
    d. Faço caso parte do desenvolvimento aconteça longe de mim
    e. Faço quando sei que passarei adiante um projeto
    f. Faço quando sei que deixarei uma equipe
    g. Faço quando posso criar um padrão de entrega
    h. Faço quando tenho que seguir um padrão de entrega estabelecido
    i. Outro

a. Sim [Questão 6]

7. Você costuma criar artefatos ou arquivos de design?
    a. Sim
    b. Não **pula para questão 10*

a. Sim [Questão 7]

11. Comparativamente, você sente que o material de design produzido por você é de melhor qualidade do que o produzido por pessoas que apenas lidam com design?

a. Sim
b. Não necessariamente
c. Não sei dizer

a. Sim [Questão 8]

12. Poderia dizer o por quê?

a. Qualidade técnica do arquivo
b. Planejamento de acordo com o desenvolvimento
c. Mindset que contempla UX e desenvolvimento
d. Outro

13. No caso de pessoas que lidam exclusivamente com design, você sente que elas deixam de fora informações relevantes para desenvolvimento? Quais seriam essas informações?

a. Não sinto que falte algo
b. Navegação entre páginas (ou seja, as páginas sequenciais e quais links levam para elas)
c. Comportamentos interativos (ou seja, a definição de feedbacks diante de um input)
d. Lógicas de fluxo (ou seja, restrições ou permissões de acesso a determinados conteúdos)
e. Os componentes visuais do layout (isto compreende: hexadecimal das cores, os padrões de botão, os padrões de tipografia, etc)
f. O embasamento para as decisões de design que foram tomadas (como pesquisas)
g. O estabelecimento de padrões visuais e/ou abstratos de funcionamento
h. O estabelecimento de exceções visuais e/ou abstratos de funcionamento

15. Na sua opinião, quais dentre esses problemas são os que mais atrapalham ou impedem um bom controle sobre um projeto? Selecione no máximo 5 opções, por favor

a. A dificuldade de comunicação
b. Ter visibilidade do impacto de mudanças
c. Retrabalho
d. Perda de informação durante o processo
e. Falta de padrão nas entregas
f. Cada alteração ter que ser comunicada e validada com diferentes partes
g. Falta de um histórico de projeto claro
h. Acesso fácil à informação
i. Organização e compreensão das informações
j. Incompatibilidade ou diferenças entre softwares
k. Não conseguir acompanhar as demandas
l. Outro

16. Quais dessas ferramentas você usa durante o seu trabalho?

a. Wordpress
b. Squarespace
c. Wix
d. Webflow
e. GitHub
f. Sketch
g. Adobe XD
h. Photoshop
i. Figma
j. Zeplin
k. InVision
l. Protopie
m. Axure
n. Principle
o. Abstract

p. Visual Studio
q. Outro

17. Como, na sua opinião, você acha que suas ferramentas poderiam te ajudar? Selecione no máximo 5 opções, por favor

a. Tendo uma interface e interações simples e intuitivas
b. Oferecendo funções voltadas tanto para design quanto para desenvolvimento
c. Me permitindo trabalhar colaborativamente
d. Tendo inteligência de versionamento
e. Me ajudando a organizar melhor meu projeto e suas informações
f. Possuindo funções que automatizem o trabalho
g. Se integrando com minhas outras ferramentas
h. Sendo de fácil acesso/compartilhamento
i. Me dando liberdade criativa
j. Funcionando como um guia
k. Funcionando com um centralizador para os meus trabalhos
l. Outro

18. Você conhece/ouviu falar no vocabulário visual de Jesse James Garrett?

a. Sim, ouvi falar, mas não uso
b. Sim, eu uso
c. Não, nunca ouvi falar

19. Gostaria de fazer algum comentário final? *Discursiva

APÊNDICE B - Roteiro da entrevista semi-estruturada para designers UX

Observação: Os colchetes funcionam como marcações para as perguntas, onde foram acrescentadas observações; as partes escritas em *itálico* são subquestões para a aprofundamento do assunto e, antecipando possíveis contratempos, as questões mais importantes foram destacadas com um asterisco (*)

1. Poderia me dizer com o que você trabalha hoje e há quanto tempo você exerce essa função?

2. Quais são suas responsabilidades e etapas de trabalho?

3. Que tipo de informações você lida para fazer o seu trabalho?

4. Como é sua experiência para conseguir essas informações? Onde você procura? Quem você consulta?
[Complementar] Que tipo de informação é difícil de achar?
O que você faz quando não consegue encontrar essas informações?

5. **[Para arquitetos]**: Onde que a Arquitetura da Informação se aplica dentro do campo de Experiência do Usuário? Onde que AI aparece no seu trabalho?

6. Através do seu processo de trabalho, você cria documentos que são usados por outras pessoas ou departamentos?
[Sim] Geralmente que tipo de documentos são produzidos?
Por quem eles são usados?
Você os adequa ao público que eles são destinados?
Por que são produzidos tipos diferentes?
[Não] Por que não?

*7. Você poderia me falar sobre o ciclo de vida desses documentos?
[Caso não fique claro]: Como é a criação, manutenção e distribuição destes documentos?
Qual ou quais etapas são as mais difíceis? Por quê?

8. O que é documentação na sua opinião?

*9. **[Condicionado a questão 3]** O que esta documentação registra de importante?
[Significando] Que tipo de informação é importante registrar?

10. Você sente que algumas informações faltam na documentação ou que são mais difíceis de serem documentadas?

[Sim] ***[Dependendo da questão 9]*** *Você diria que a documentação acaba sendo mais voltada para as qualidades visuais?*
Faz sentido a produção de uma documentação focada em arquitetura da informação? Por quê?
[Não] Por quê? Como que você faz para isso dar certo?

11. Qual o valor em se produzir esses documentos? Por quê?
[Significando] Para o que esses documentos servem?

12. **[Dependendo do tempo da entrevista]** Quais ferramentas você utiliza hoje? Você sente que elas conseguem suprir suas necessidades de documentação? E quanto à arquitetura da informação?
[Sim] O que elas têm de especial?
[Não] Poderia dizer por que não? O que falta nelas?

*13. Considerando o material entregue por UX e o que foi desenvolvido, o que você diria que é a documentação final: o material de UX ou o código?
[Caso eu sinta que o entrevistado consegue falar sobre isso] Como fica a documentação de design no caso do código ser diferente da documentação?

14. A organização em que você trabalha adota algum tipo de metodologia de trabalho? O que essa metodologia tem de bom e o que ela tem de ruim?
[Sugerir que o entrevistado fale sobre] Especificamente relação UX-DEV
Hand over de UX para DEV e entre UXs

*15. Quais são as maiores dificuldades (pain points) de lidar com esses documentos dentro desta metodologia?
[Aqui eu espero coisas como]: Desafios de workflow; Dificuldade em registrar informações; Benefícios da automatização de inspect; Divergências interpretativas sobre a documentação
[Sugerir para o entrevistado] Faz sentido produzir uma documentação sob essas condições? Por quê?
Qual é o peso da metodologia sobre a produção?
Faz sentido manter a documentação atualizada na medida em que ela é executada em código?

*16. Você acha que essa maneira de trabalho adotada pela sua empresa é contraditória ao mindset de design?
[Buscar por informações relacionadas a] Visibilidade do projeto; Sustentabilidade do sistema, a perda sobre o domínio da estrutura, a perda sobre a visibilidade de desenvolvimento

17. Se você pudesse alterar três coisas sobre como documentações de design são feitas hoje, o que você alteraria?

18. Gostaria de fazer algum comentário, observação ou sugestão?

APÊNDICE C - Roteiro da entrevista semi-estruturada para desenvolvedores

Observação: Os colchetes funcionam como marcações para as perguntas, onde foram acrescentadas observações; as partes escritas em *itálico* são subquestões para a aprofundamento do assunto e, antecipando possíveis contratempos, as questões mais importantes foram destacadas com um asterisco (*)

1. Poderia me dizer com o que você trabalha hoje e há quanto tempo você exerce essa função?

2. Quais são suas responsabilidade e etapas de trabalho?
[Caso o entrevistado não mencione]: Quais ferramentas você utiliza?

*3. Qual você diria que é o papel do design dentro do seu workflow?

*4. Que tipo de informações você precisa para fazer o seu trabalho?
[Significando] Na sua opinião, quais são os componentes básicos de um projeto de UX que desenvolvimento deveria ter visibilidade?

5. Quais materiais você geralmente recebe de UX? Por que eles são importantes?

*6. Você fica geralmente satisfeito com a documentação? Por quê?
[Espero informações relacionadas à]: Maneira como chega; Eficiência; Maneira como é registrada

7. **[Dependendo de como seja respondida a questão 6, pode pular]** Quais são os seus pain points em relação à uma documentação de UX?
[Espero informações relacionadas à]: Desafios de workflow; Dificuldade em consumir (e questionar) informações para desenvolvimento; Benefícios da automatização de inspect; Divergências interpretativas sobre a documentação

8. **[Caso este não tenha sido um pain point]** Como que esta documentação chega geralmente até você?

[Espero informações relacionadas à]: Ferramentas; Formatos de documentação; Dinâmicas de trabalho; Este contato é fácil ou difícil?

9. Quais ferramentas você utiliza hoje? Você sente que elas conseguem suprir suas necessidades de documentação?

*10. **[Dependendo da resposta da questão 6]** O que deveria constar em uma documentação na sua opinião? Falta alguma informação?

*11. Considerando a entrega final (o fim de um projeto), o que você diria que é a documentação final: o material de UX ou o código?
[Caso eu sinta que o entrevistado consegue falar sobre isso] Como fica a documentação de design no caso do código ser diferente da documentação?

12. A organização em que você trabalha, adota algum tipo de metodologia de trabalho? Qual? Como ela é?

13. O que essa metodologia tem de bom e o que ela tem de ruim?
[Sugerir que o entrevistado fale sobre] Especificamente relação UX-DEV
Hand over de UX para DEV e entre UXs

*14. Quais são as maiores dificuldades de se lidar com UX dentro de uma organização que adota esta metodologia?

15. Tem alguma coisa que você gostaria que os designers soubessem ou tivessem em mente no momento de produzir esses materiais?

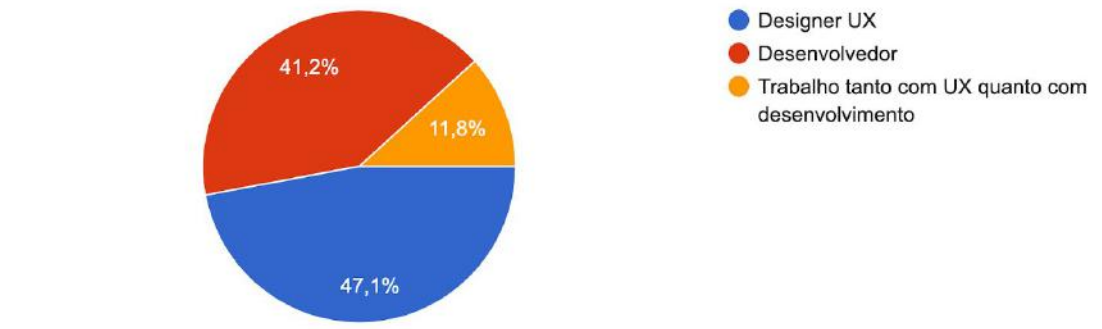
16. Se você pudesse alterar três coisas sobre como documentações de design são feitas hoje, o que você alteraria?

17. Gostaria de fazer algum comentário, observação ou sugestão?

APÊNDICE D - Respostas do questionário online

Qual é a função que você desempenha hoje?

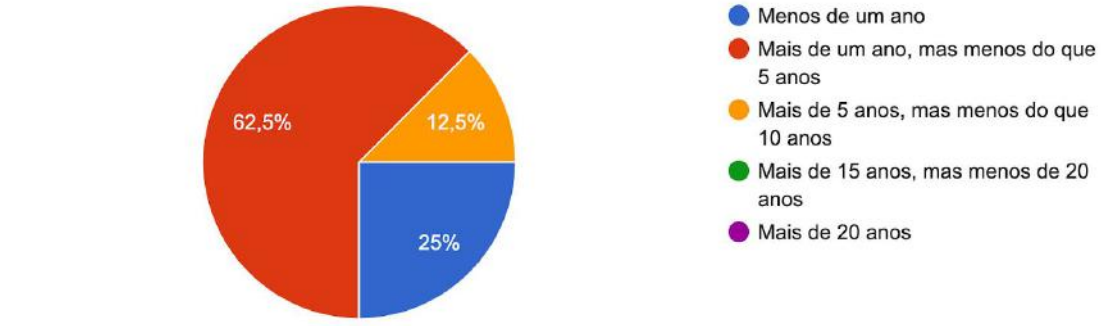
17 respostas



Designers UX

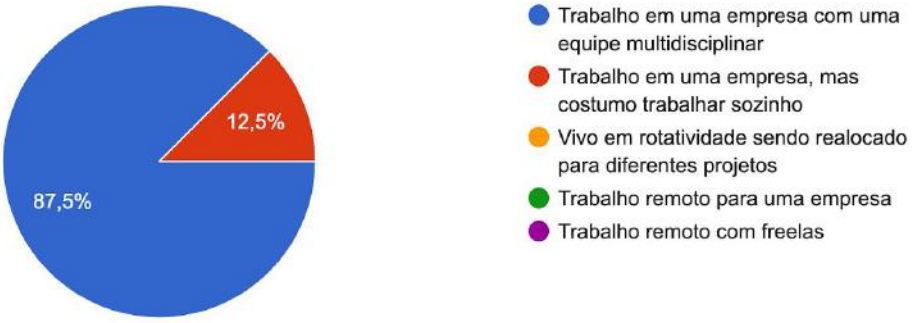
Há quanto tempo você trabalha nessa área?

8 respostas



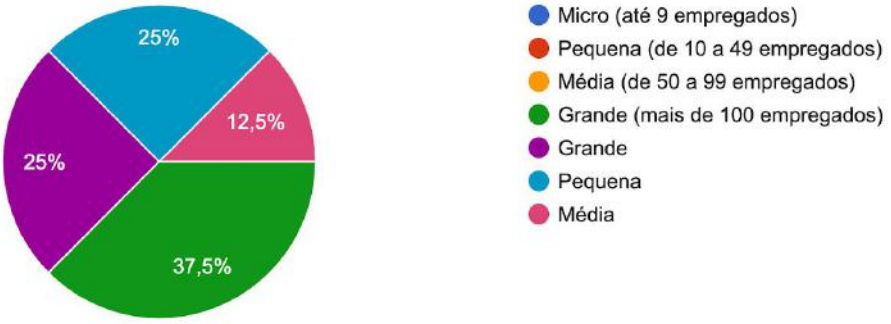
Onde que você trabalha hoje?

8 respostas



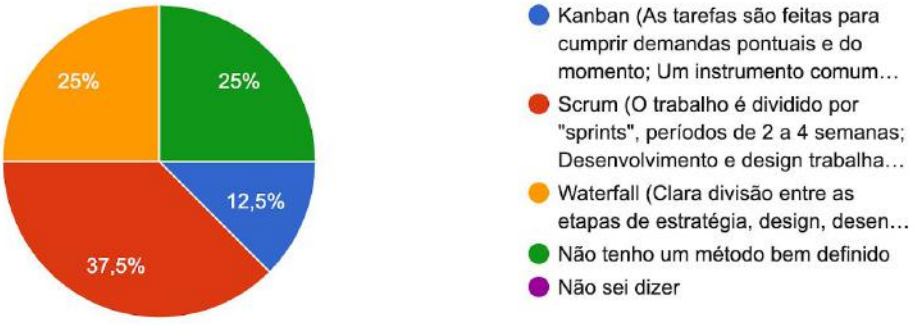
Qual o porte da empresa em que você trabalha? (Mesmo que você trabalhe em uma filial, por favor, considere o tamanho de toda a empresa)

8 respostas



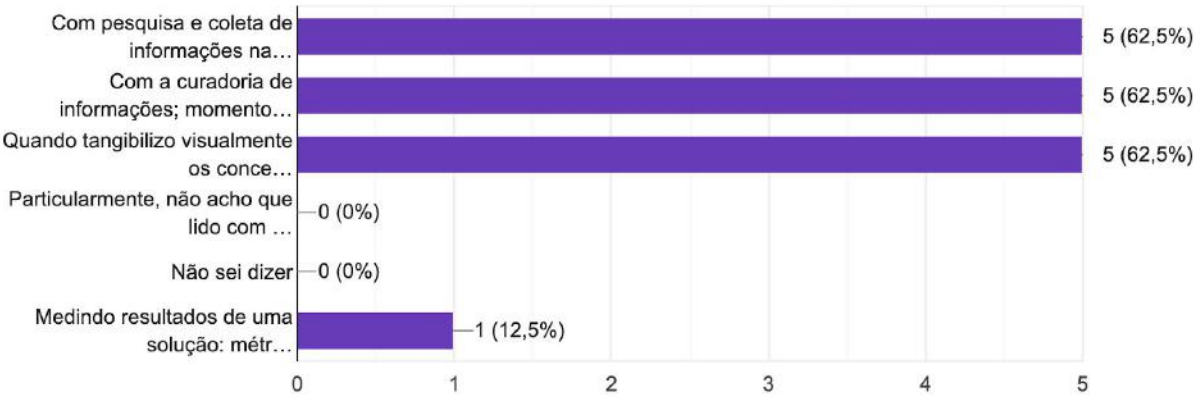
Como que costuma ser seu método de trabalho?

8 respostas



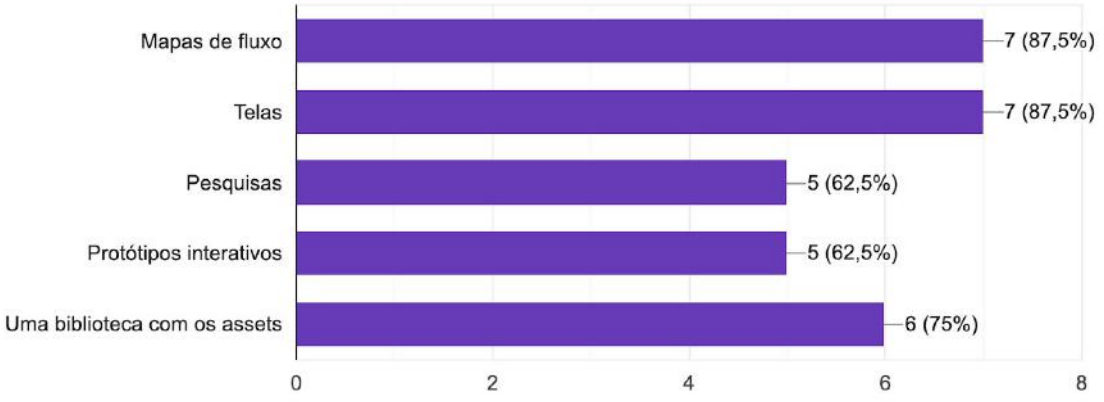
Como que o trato de informações costuma aparecer durante o seu fluxo de trabalho?

8 respostas



Quais materiais você costuma produzir?

8 respostas



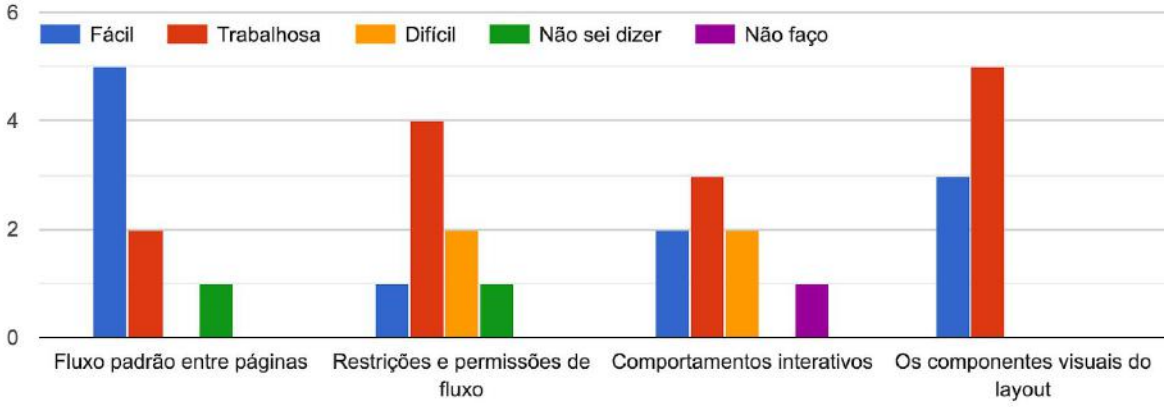
Poderia dizer por que você recorre especificamente a esses materiais? *OPCIONAL

2 respostas

Eles fazem parte do processo. Mapas de fluxo, telas, levantamento de pesquisa teórica pras decisões, testes e pesquisas de usuário são a parte investigativa que me leva a desenvolver protótipos interativos e assets. Acredito que recorro a eles por serem a maneira mais divulgada de fazer um projeto de interação digital, nunca procurei outras formas de conceber e depois colocar em prática.

porque acho mais prática a visualização do macro

A partir desses materiais, como é sua experiência para registrar as seguintes informações?



Existe mais algum tipo de informação que você sente dificuldade em transmitir visualmente? *OPCIONAL

3 respostas

Escolhas gráficas que vêm da minha experiência pessoal, e não de referências digitais ou de pesquisa. Após anos estudando design, eu tenho alguns conhecimentos teóricos que fazem da experiência/layout melhor, mas não sei explicar nem citar fontes confiáveis. Tenho medo de perder a credibilidade nas minhas escolhas gráficas por falta de argumentação.

Métricas

Eventuais mudanças na interface (garantir que o documento consultado será sempre o mais atual).

Você poderia dizer por que acha que são informações difíceis de transmitir? *OPCIONAL

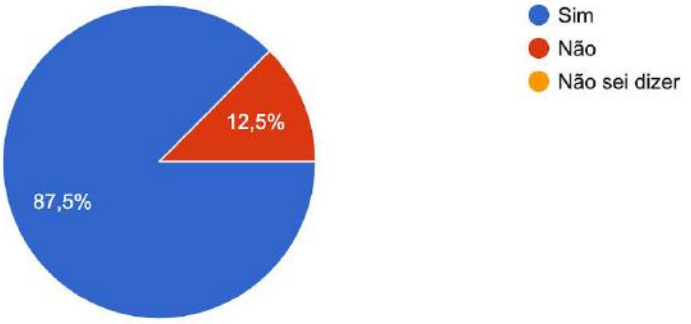
2 respostas

Não há fontes formais e as escolhas podem parecer subjetivas se não forem explicadas/defendidas

É difícil mostrar como funcionará o tracking de eventos e como isso vai se refletir posteriormente

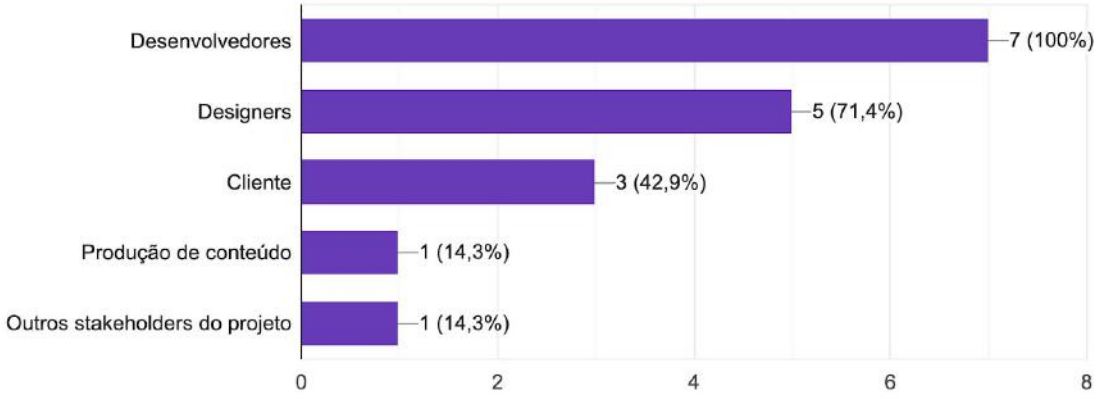
Esses materiais produzidos por você são usados por outras equipes ou departamentos?

8 respostas



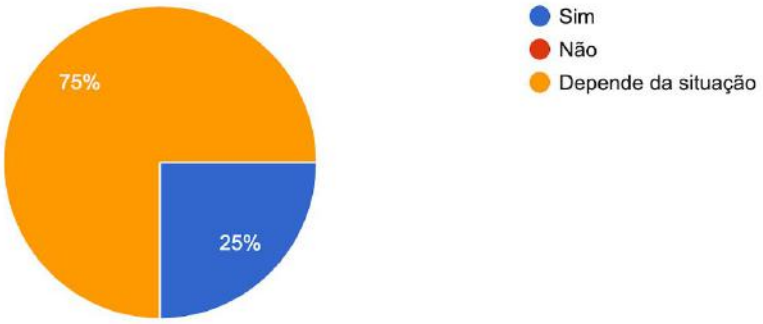
Para quem eles são destinados?

7 respostas



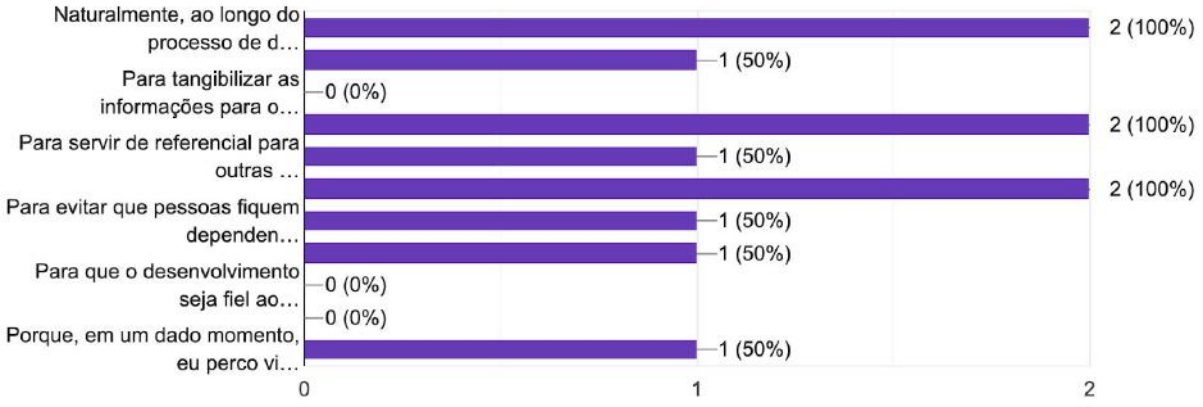
Você costuma fazer documentação?

8 respostas



Por que você costuma fazer documentação?

2 respostas



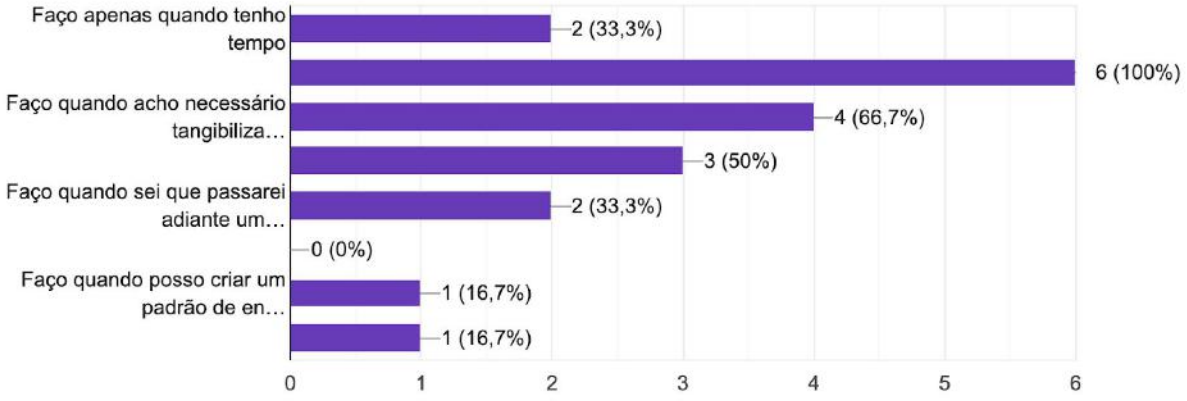
Por quê você não costuma fazer documentação?

0 resposta

Ainda não há respostas para esta pergunta.

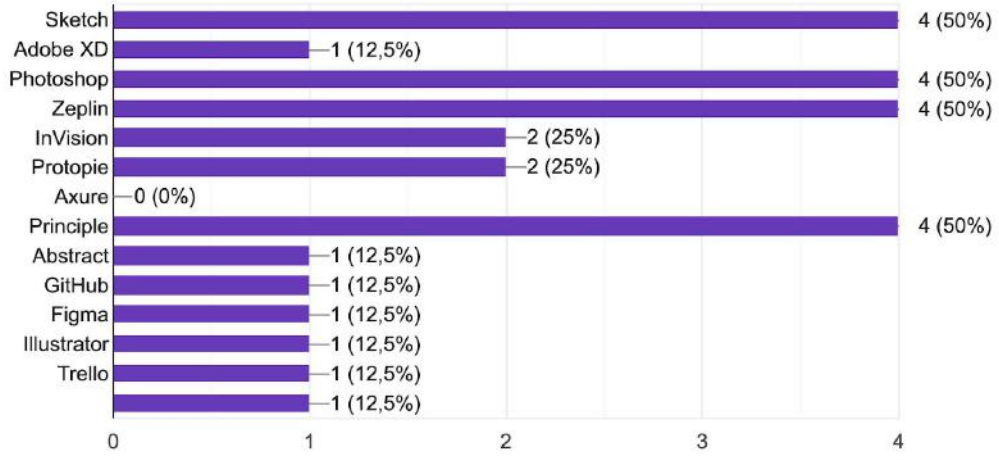
Sob quais situações você faz documentação?

6 respostas



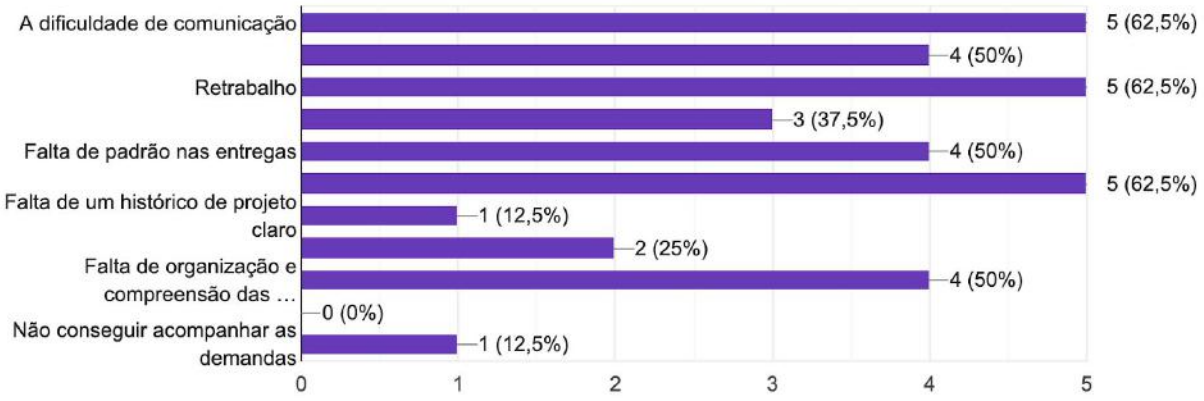
Quais dessas ferramentas você usa durante o seu trabalho?

8 respostas



Na sua opinião, quais dentre esses problemas são os que mais atrapalham e impedem um bom control...ione no máximo 5 opções, por favor

8 respostas

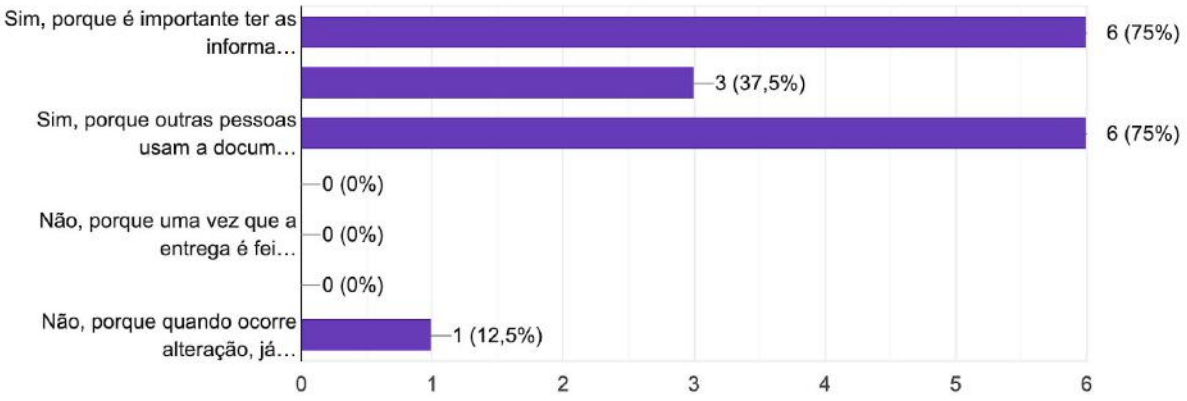


Como, na sua opinião, você acha que suas ferramentas poderiam te ajudar? Selecione no máximo 5 opções, por favor

8 respostas

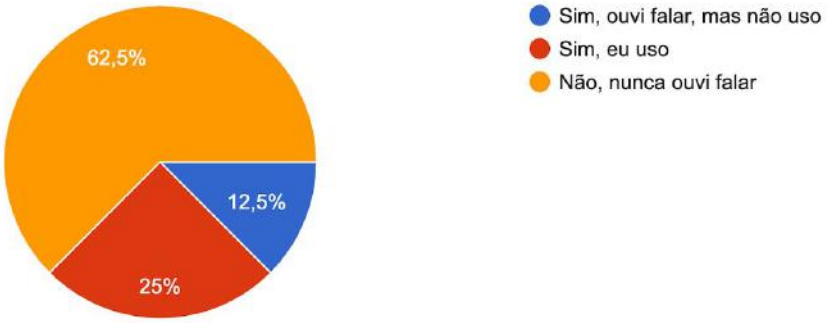
Definições de projeto mudam mesmo durante o desenvolvimento. Uma vez o projeto está sendo codado, você ...ido atualizar os materiais de design?

8 respostas



Você conhece o vocabulário visual de Jesse James Garrett?

8 respostas



Gostaria de fazer algum comentário final sobre documentação? *OPCIONA

1 resposta

Não

Desenvolvedores

Há quanto tempo você trabalha nessa área?

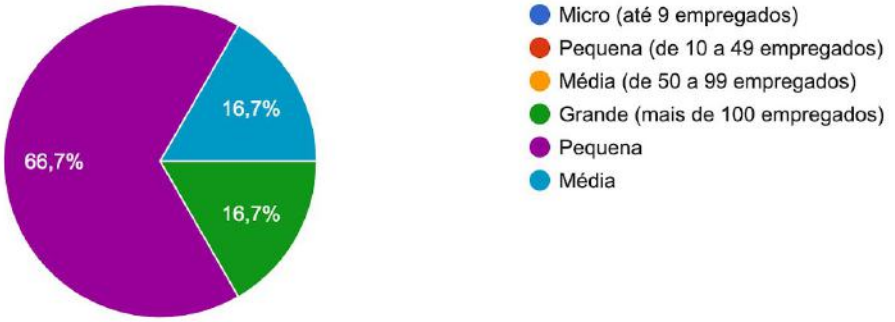
7 respostas

Onde que você trabalha hoje?

7 respostas

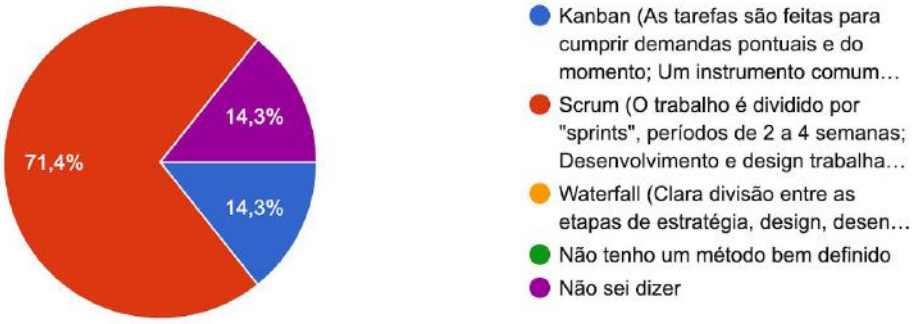
Qual o porte da empresa em que você trabalha? (Mesmo que você trabalhe em uma filial, por favor, considere o tamanho de toda a empresa)

6 respostas



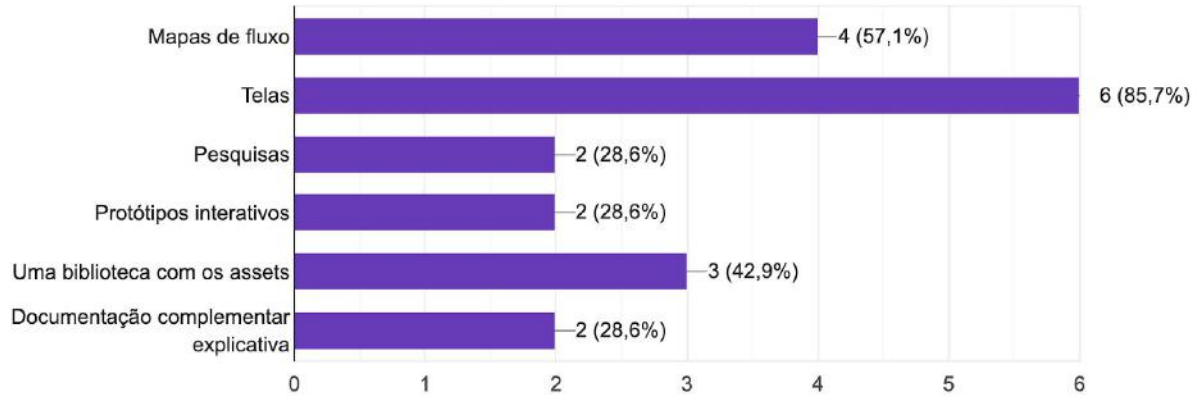
Como que costuma ser seu método de trabalho?

7 respostas



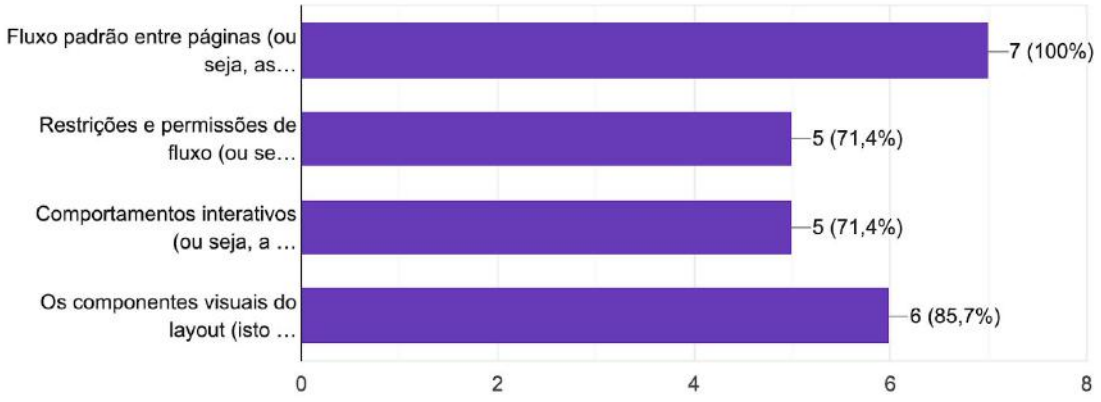
Quais materiais você considera parte de uma entrega mínima, isto é, que são indispensáveis para que você consiga desenvolver?

7 respostas



A partir desses materiais, quais informações sobre o design você precisa ter para uma entrega mínima?

7 respostas



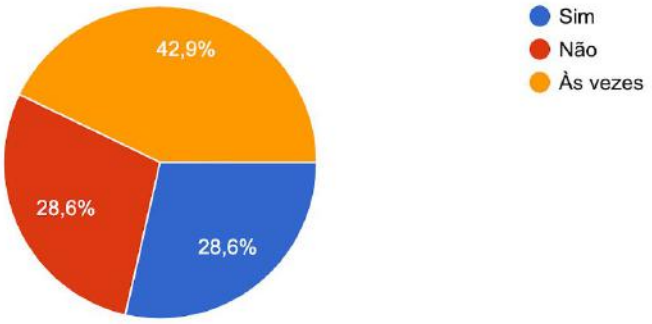
Existe mais algum tipo de informação (mais abstrata ou visual) que você sente que as entregas de design não contemplam? *OPCIONAL

1 resposta

A inspiração do próprio designer

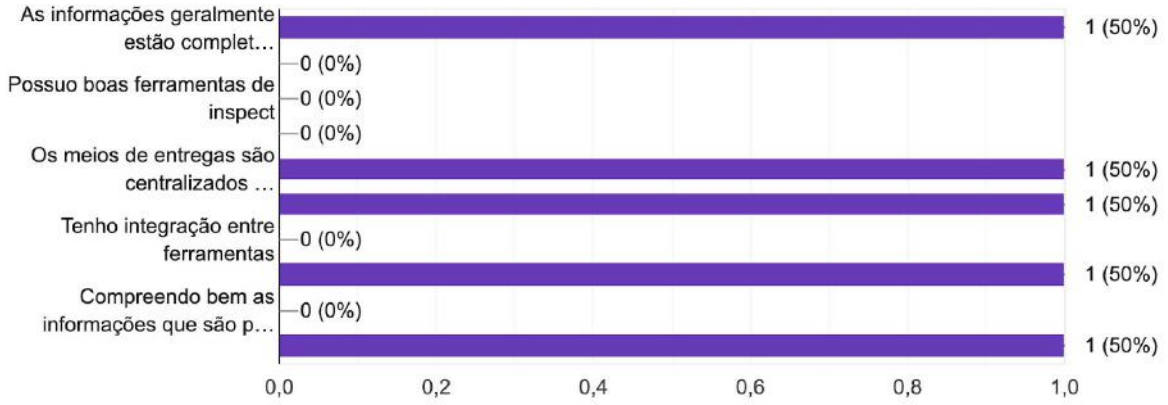
Você fica satisfeito com os materiais de design?

7 respostas



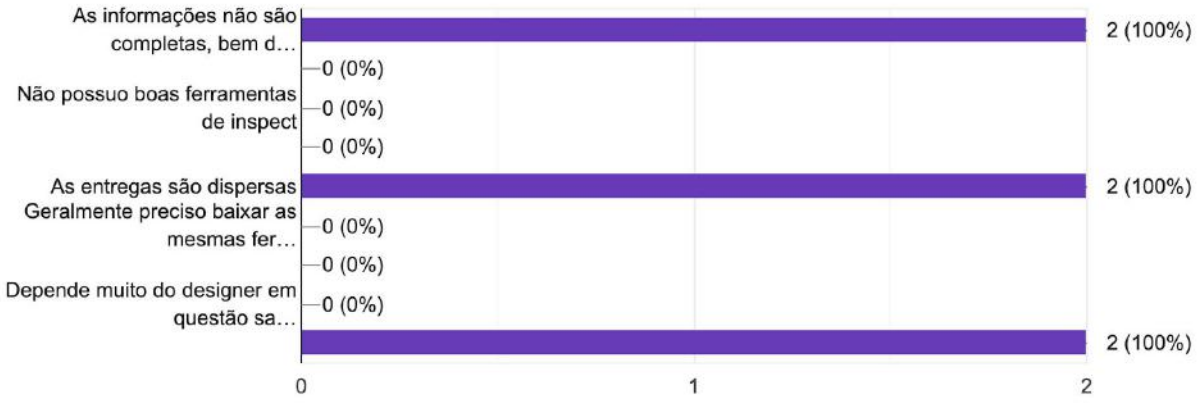
Por que você fica satisfeito?

2 respostas



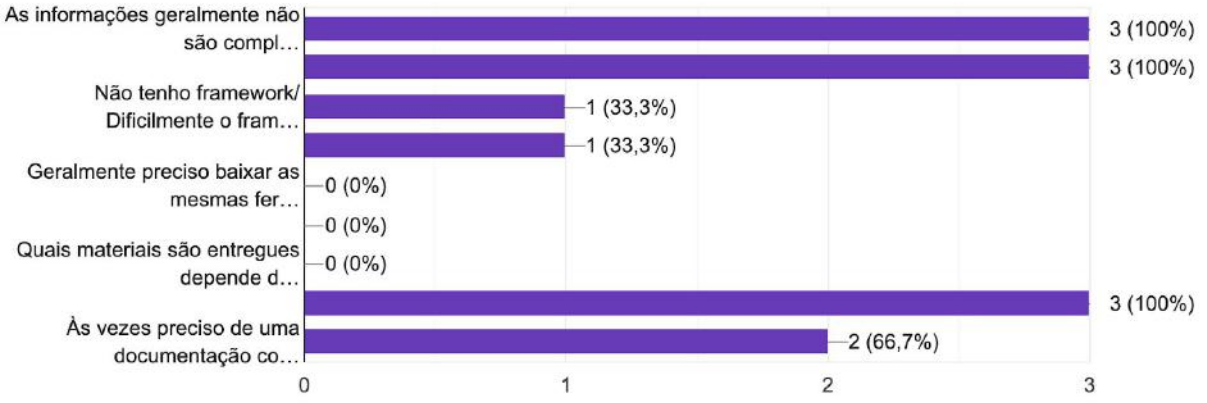
Por que você não fica satisfeito?

2 respostas



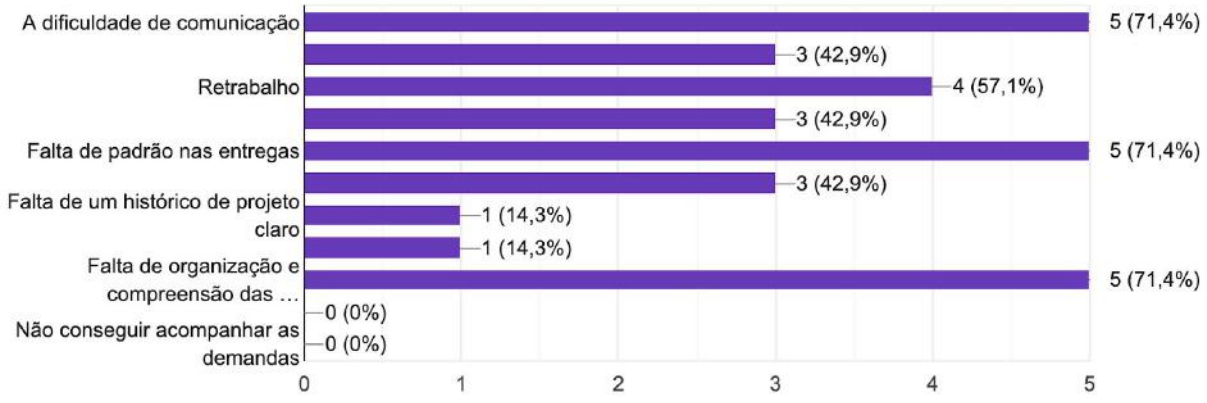
Por que você fica satisfeito com esses materiais apenas às vezes?

3 respostas



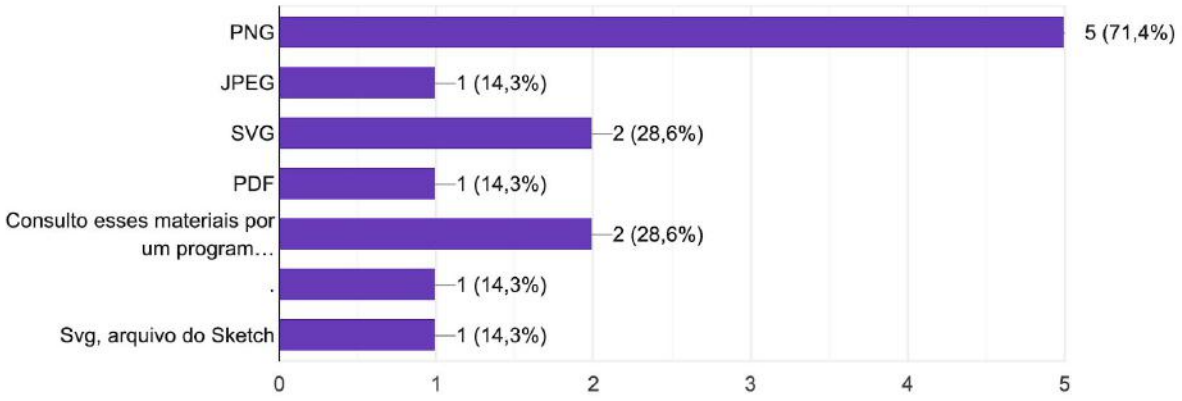
Na sua opinião, quais dentre esses problemas são os que mais atrapalham e impedem um bom control...ione no máximo 5 opções, por favor

7 respostas



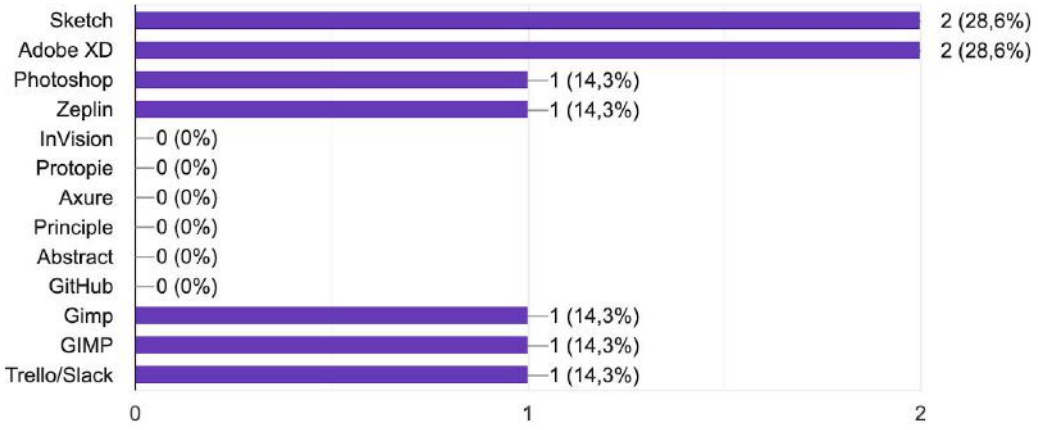
Geralmente esses materiais são disponibilizadas para você em quais formatos?

7 respostas



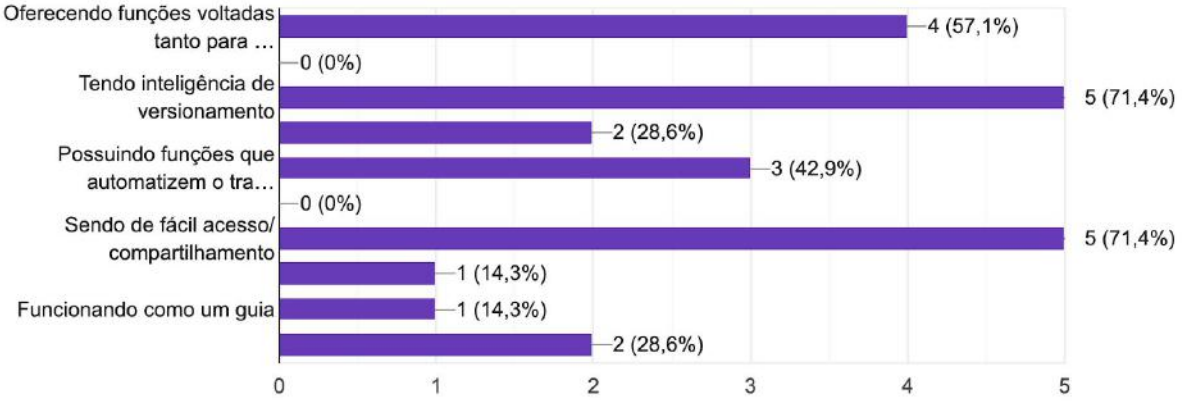
Quais dessas ferramentas você usa para consultar os materiais de design?

7 respostas



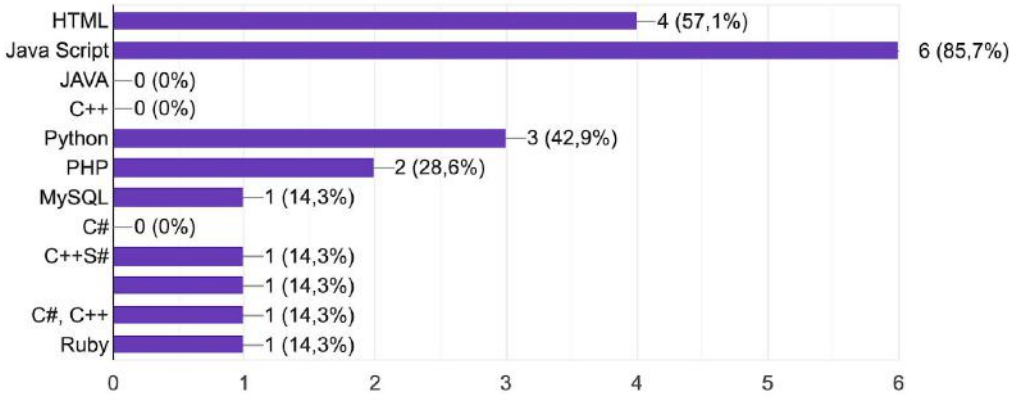
Como, na sua opinião, você acha que suas ferramentas poderiam te ajudar? Selecione no máximo 5 opções, por favor

7 respostas



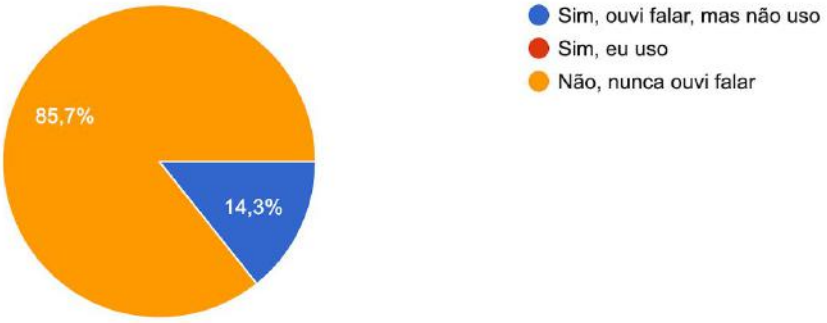
Quais são as “linguagens” de programação mais usadas por você?

7 respostas



Você conhece o vocabulário visual de Jesse James Garrett?

7 respostas



Gostaria de fazer algum comentário final sobre documentação? Algo que você gostaria que designers soubessem e levassem em consideração no momento de fazer seus materiais? *OPCIONAL

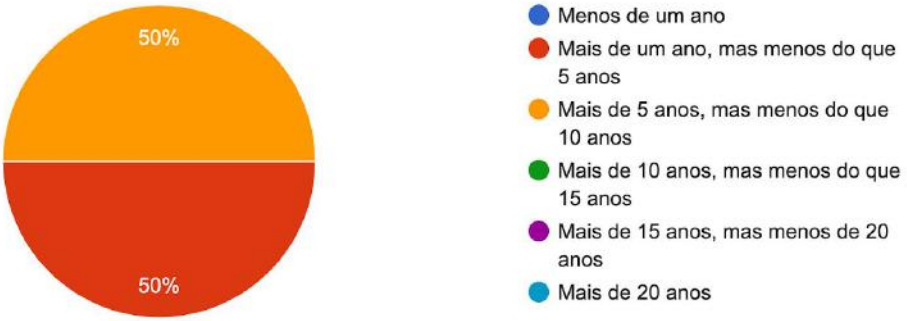
3 respostas

Disponibilizar um documento ou protótipo ilustrando as interações na tela, os fluxos de telas e regras de exibição. Atualmente só me disponibilizam um arquivo sketch e o restante é todo à base da conversa, que nem sempre flui tão bem e retarda o desenvolvimento/causa retrabalho pela constante descoberta de fluxos não previstos)
Padrozinação é vida
Não focar apenas no projeto estático, imaginar fluxos neste projeto e pensar em meios de quebrar o fluxo do seu projeto, assim poderá antecipar alguns bugs evitando que o mesmo seja descoberto durante o desenvolvimento evitando retrabalho.

Designers UX + Desenvolvedores

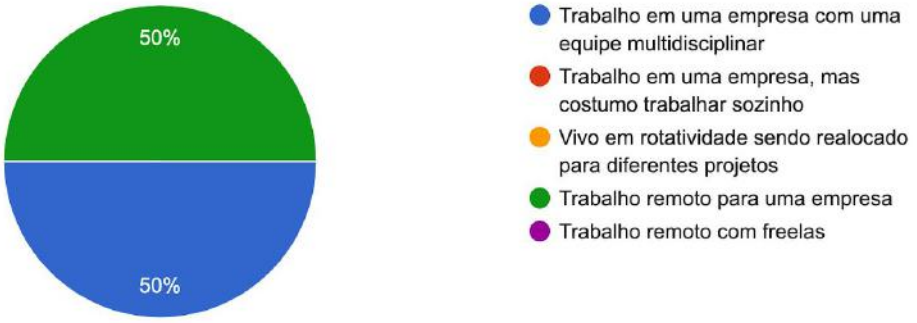
Há quanto tempo você trabalha nessa área?

2 respostas



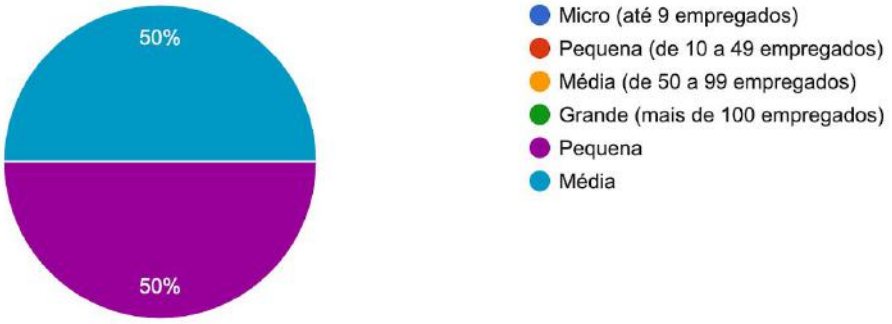
Onde que você trabalha hoje?

2 respostas



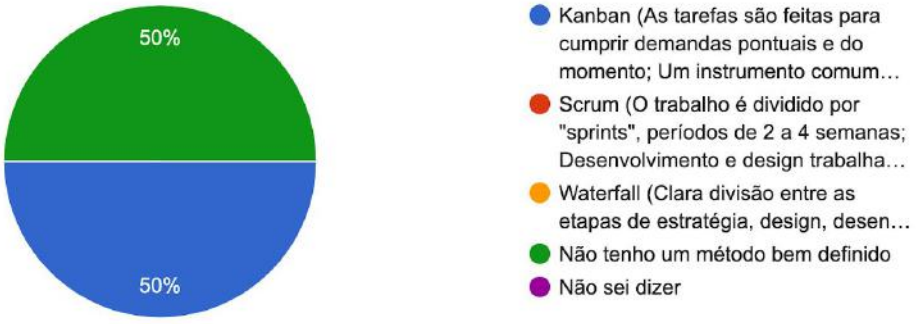
Qual o porte da empresa em que você trabalha? (Mesmo que você trabalhe em uma filial, por favor, considere o tamanho de toda a empresa)

2 respostas



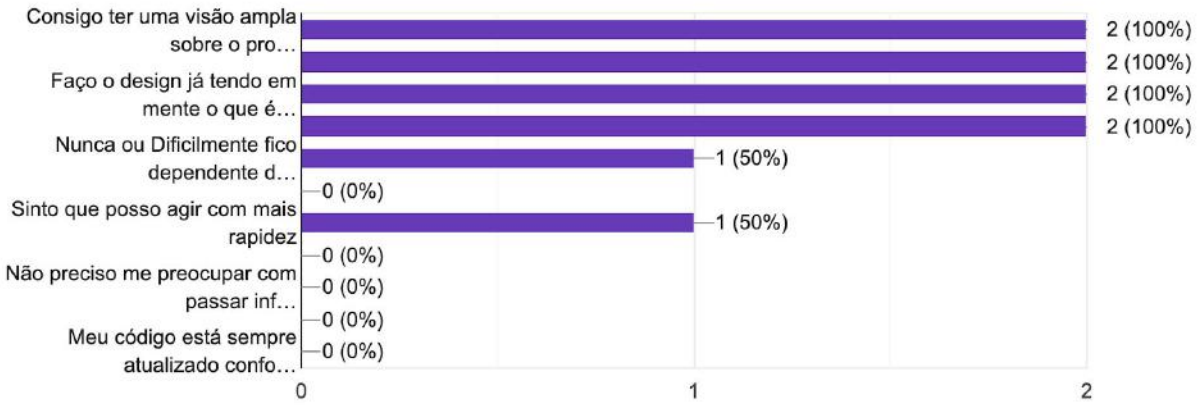
Como que costuma ser seu método de trabalho?

2 respostas



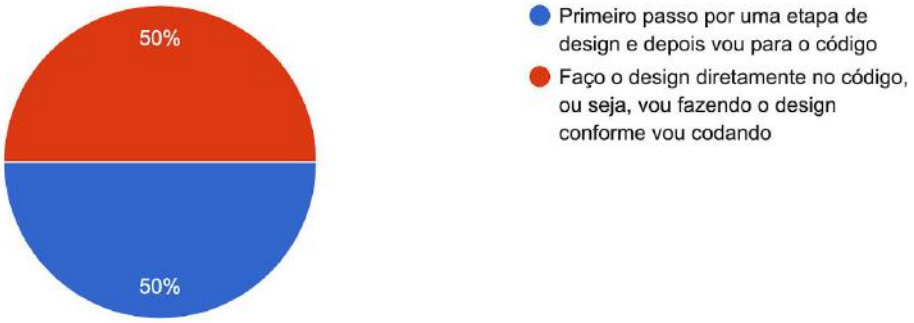
Quais são as 5 maiores vantagens de se lidar com esses extremos (design e código) no seu trabalho?

2 respostas



Qual o método de trabalho que você adota com mais frequência?

2 respostas



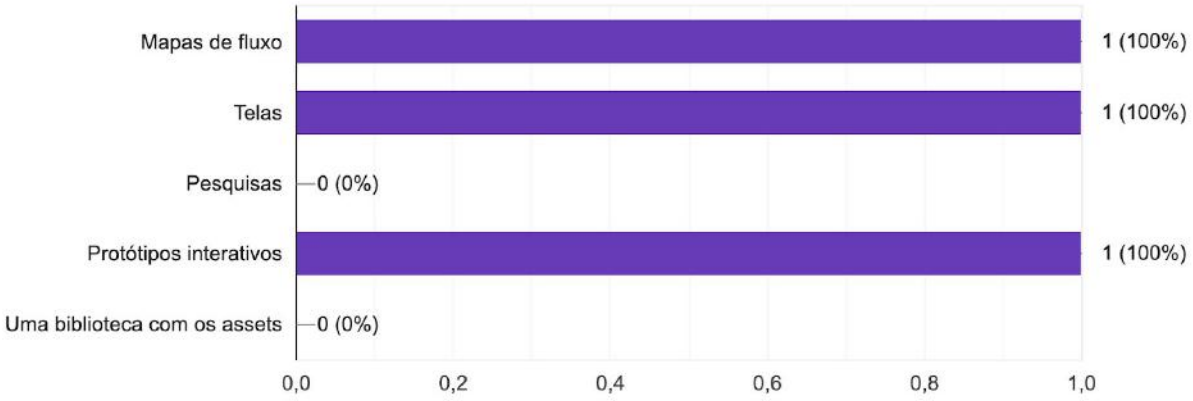
Poderia dizer por que você recorre especialmente a esses materiais? *OPCIONAL

0 resposta

Ainda não há respostas para esta pergunta.

Quais materiais você geralmente produz?

1 resposta



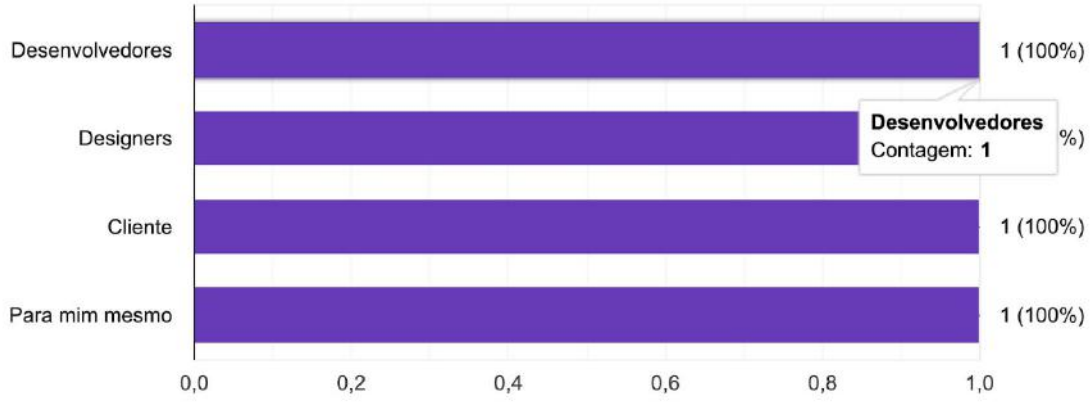
Poderia dizer por que você recorre especialmente a esses materiais? *OPCIONAL

0 resposta

Ainda não há respostas para esta pergunta.

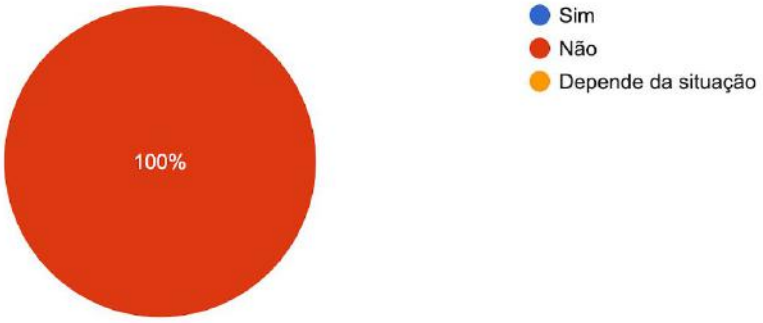
Pode nos dizer para quem eles são destinados?

1 resposta



Você costuma fazer documentação?

1 resposta



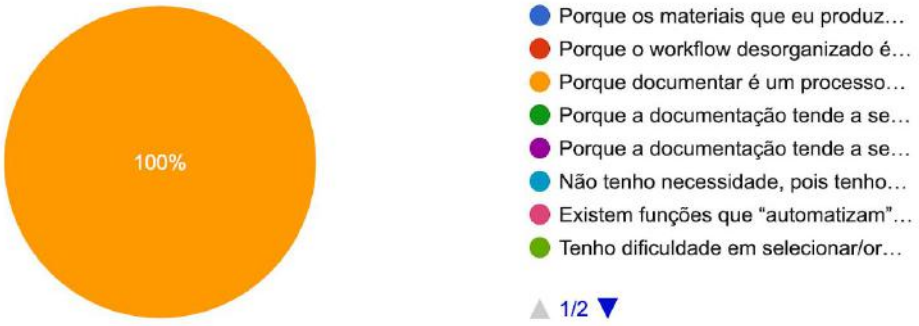
Por que você costuma fazer documentação?

0 resposta

Ainda não há respostas para esta pergunta.

Por quê você não costuma fazer documentação?

1 resposta



Sob quais situações você faz documentação?

0 resposta

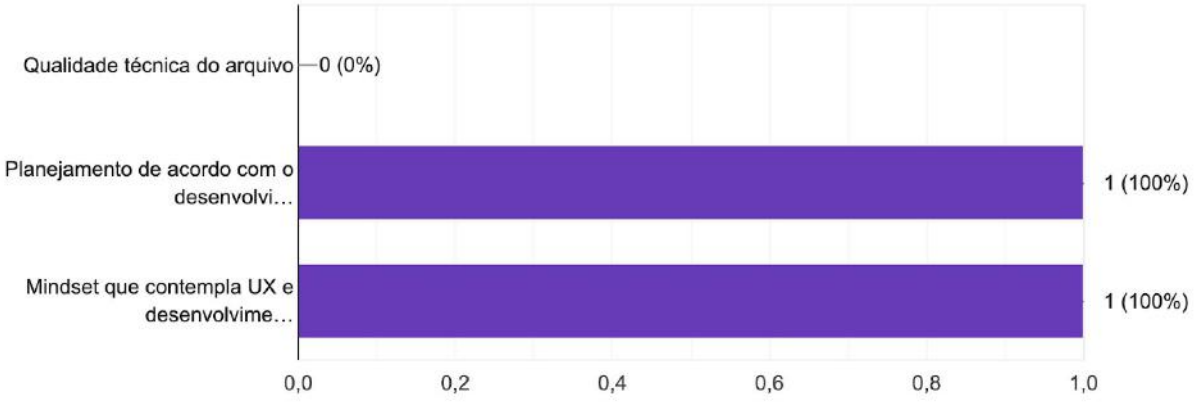
Ainda não há respostas para esta pergunta.

Comparativamente, você sente que o material de design produzido por você é de melhor qualidade do que o...essoas que apenas lidam com design?

1 resposta

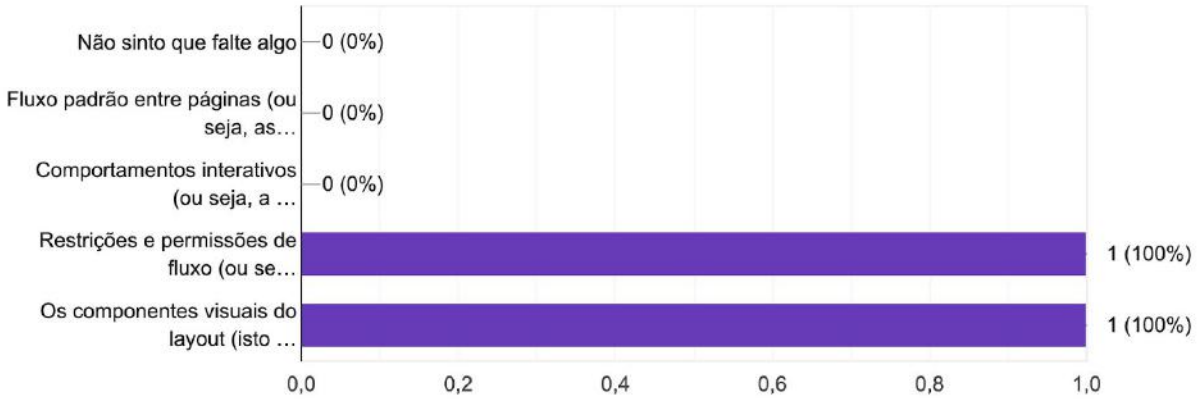
Poderia dizer o por quê? *OPCIONAL

1 resposta



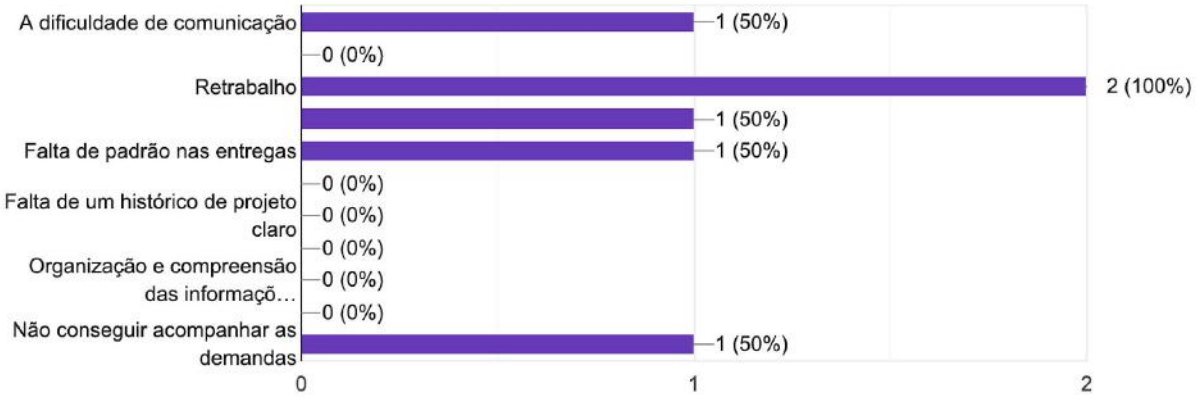
No caso de pessoas que lidam exclusivamente com design, você sente que elas deixam de fora informações relevantes para desenvolvimento?

1 resposta



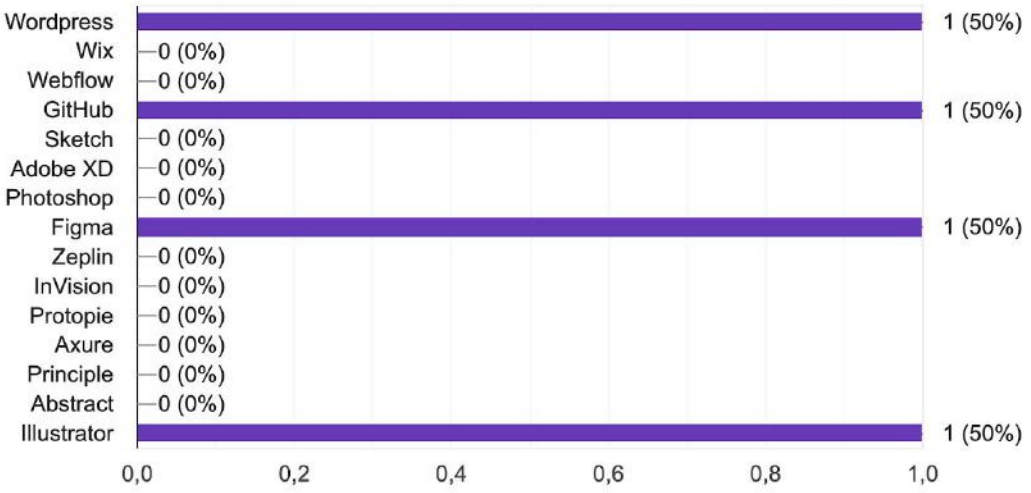
Na sua opinião, quais dentre esses problemas são os que mais atrapalham e impedem um bom control...ione no máximo 5 opções, por favor

2 respostas



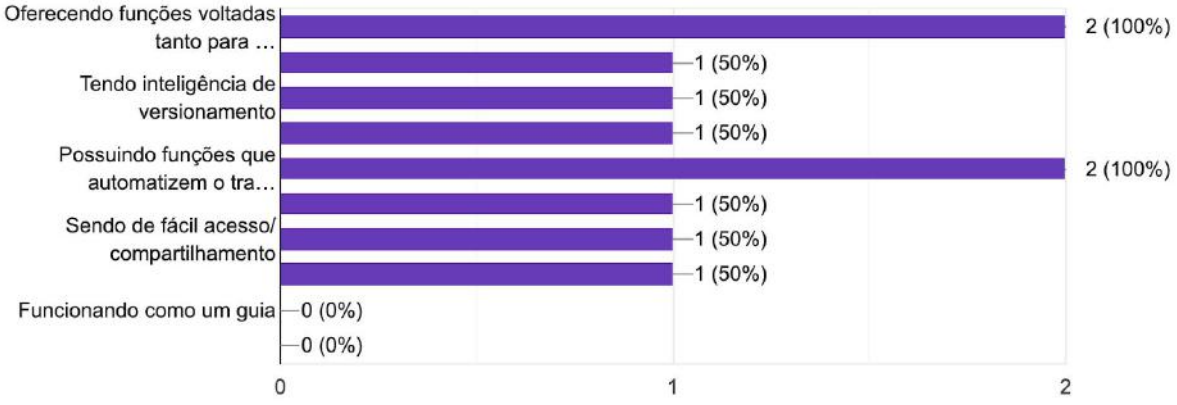
Quais dessas ferramentas você usa durante o seu trabalho?

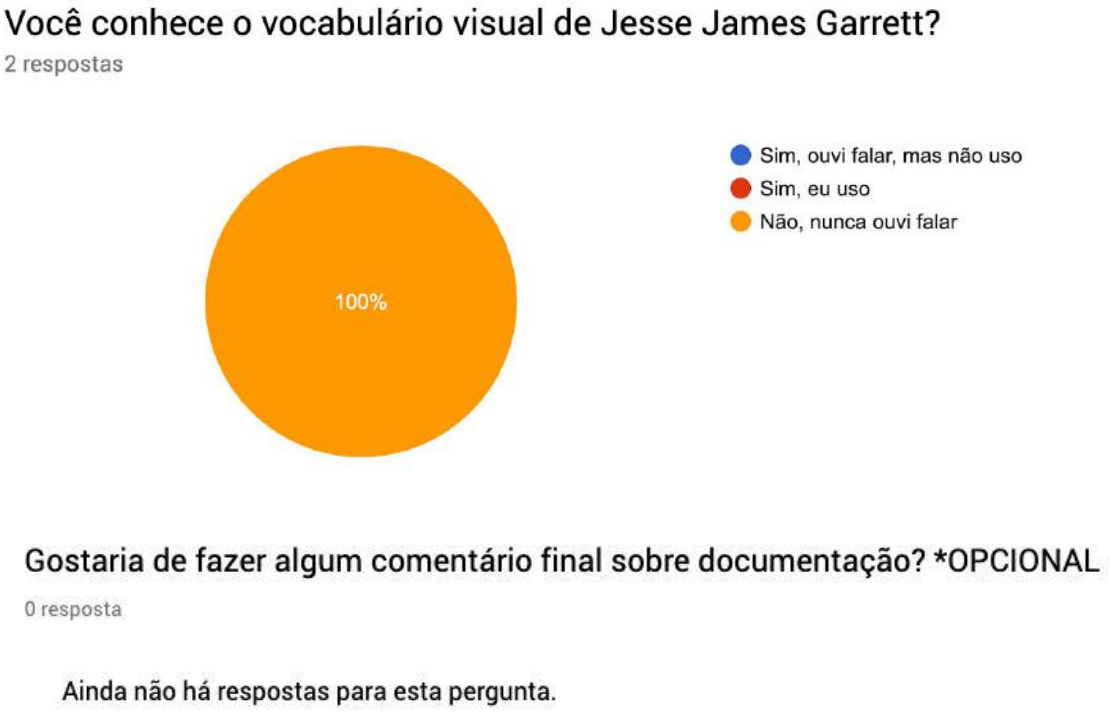
2 respostas



Como, na sua opinião, você acha que suas ferramentas poderiam te ajudar? Selecione no máximo 5 opções, por favor

2 respostas





APÊNDICE E - Respostas das entrevistas semi-estruturadas para designers

UX1

E: Poderia me dizer com o que você trabalha hoje e há quanto tempo você exerce essa função?

UX 1: Como designer, eu trabalho desde 99… Primeiro *freelancer*[50], depois estágio e agências. Mas me reconhecer como *UX*, foi mesmo aqui, onde eu entrei em 2006… em 2007, que aconteceu essa mudança cultural aqui dentro… Das pessoas entenderem o que o mercado queria, a gente se reconhecer como *UX* designers mesmo.

[50] Trabalhador independente, autônomo

E: Quais são suas responsabilidades e etapas de trabalho?
UX 1: Como qualquer designer, tem uma parte de entendimento de qual é o problema, qual é a questão a ser solucionada; qual a demanda sob dois pontos de vista: O da empresa, como que isso convive com os meus interesses, enquanto empresa, e o ponto de vista do usuário, qual o desejo dele, qual é o problema do ponto de vista do usuário e, daí, em cima de essa interseção de *user needs*[51] e *business goals*[52], a gente começa a investigar. Desde, dependendo do tamanho do projeto, a gente consegue fazer entrevistas com usuários, se familiarizar com casos de uso, categorizar usuários, entender mesmo como que seria mesmo as pessoas usando, até ler documentação sobre o mercado, métricas atuais, se é um produto que já existe e se você está trabalhando com uma melhoria, literatura, *papers*… Então, eu acho que tem uma primeira etapa mesmo de se munir de informações a respeito do problema e às vezes até ajudar redefinir o problema, do tipo "Chegou aqui a demanda, mas após aqui o meu entendimento, vi que não é bem isso". Se isso é um problema dum problema maior, se esse é um problema mesmo, mas se estava sendo visto de um ponto de vista torto. E uma fase de geração de ideias e tem uma série de técnicas para gerar possibilidades. Depois tem uma seleção sobre as ideias que se mostram mais viáveis, quais podem emplacar, quais são as mais vantajosas naquele cenário; essas que tu acredita mais, tu pode chegar ao ponto de prototipa-las e fazer um teste mais rápido antes de desenvolvê-la no total e até o ponto de acompanhar o desenvolvimento, de fato. Às vezes quebrar essa ideia em várias fases e juntar esses incrementos até formar uma versão lançável, uma versão mínima do que seria esse produto e botar no ar com métricas, ver se está performando ou não da maneira como tu imaginou, pensar em quais seriam essas métricas de sucesso em relação a definir se está funcionando ou não o que tu projetou… E acho que depois disso é um ciclo em cima disso, né, inteirar em cima das métricas que tu coleta e, de novo, otimizar, otimizar, otimizar.

E: O que você quis dizer por viabilidade?

[51] Necessidades do usuário
[52] Objetivos de negócio

UX 1: Tem viabilidade técnica, da gente ter o *know how*[53] para saber fazer e às vezes é a viabilidade de tempo e esforço. Às vezes eu tenho um *time to marketing,*[54] em que eu tenho que estar com algo lançado, e isso, dependendo da proposta, não é viável nesse espaço de tempo, então, tem que ver a viabilidade desse espaço de tempo. Outra é "não sei se é viável" o esforço, do tipo, a gente tem o *know how*, tem tempo para fazer, mas isso requer um esforço muito grande comparado com outra solução que à princípio também atende ou atende a primeiro momento dado que a gente não sabe nem se vai funcionar a abordagem, então, não se quer fazer um esforço tão grande para uma solução tão robusta, tão complexa, e daí, se prefere ver essa viabilidade de esforço para saber se esse custo-benefício tá valendo a pena. E outra é a habilidade técnica, seja de *know how* ou de infraestrutura mesmo, de equipamento. Se eu tenho como fazer o que eu tô propondo, se é uma tecnologia que a gente não está acostumado, é todo um custo para aprender, para terceirizar...

E: Como fazer versionamento de um projeto até se chegar na versão ideal?
UX 1: Acho que é muito fácil de imaginar, pelo menos para mim, esse momento de quebra e construção em partes do projeto do ponto de vista de desenvolvimento. Então, de fato quando a gente estiver fazendo a coisa, ir fazendo ela pedaço a pedaço até um recorte mínimo que faça sentido, uma completude, um recorte que faça sentido em si, e depois, as outras fases serem incrementos que se somam ao que já existia. Do ponto de vista projetual, é um pouco mais difícil. Não quer dizer que seja impossível ou não aplicável. Eu acho que existe uma tendência, pelo menos da minha parte, de olhar para o todo para ter certeza de que não estou deixando furo, então, talvez [eu] tenha um desconforto com a ideia de "consertar posterior"; eu tento projetar o mais amplo possível dentro do espaço que me dão de tempo para minimizar o meu anseio de deixar alguma ponta solta em um sistema maior, mas de tal forma que isso não prejudique as entregas incrementais. Então, eu tento fazer o que precisa mais um pouco pensando no todo.

[53] O "saber fazer", *expertise*, competência para se fazer
[54] Tempo do produto ser concebido até ele ser lançado para venda

E: Através do seu processo de trabalho, você cria documentos que são usados por outras pessoas ou departamentos?
UX 1: Sim

E: Você poderia me falar sobre o ciclo de vida desses documentos?
UX 1: Antes de falar sobre o ciclo de vida destes documentos, acho que eu preciso dizer que documentos são esses. Acho que com a adoção do método ágil, aqui dentro, a gente puxou mais a corda muito para um lado e, dado que fazer uma mudança de cultura é muito difícil, a gente pesou muito a mão do desenvolvimento ágil, quase como se a gente estivesse andando com a bíblia do *Scrum* debaixo do braço. Que que isso implica: Lá no manifesto ágil tá lá que "A gente prioriza comunicação sobre documentação"; uma interpretação errada disso que pode ser perigosa, que eu acho que se criou aqui, é que se criou uma aversão à documentação. Documentação era, dentro daquele modelo *Design Up Front*[55]*,* eu desenhava tudo e gerava um documento que explicava aquelas interface e o funcionamento delas; então, para um desenvolvedor que não tinha contato com o designer, [ele] tinha que pegar as interfaces, pegar o manual e ler aquilo e essa era minha interface, essa era minha comunicação, era um documento; isso é pobre, né, dentro de um contexto, comparado a desenvolvimento ágil, onde eu tenho uma mini-equipe super comunicativa. Realmente, olhar para um documento e é ser só ele que eu posso consultar, é muito pobre. Então, se criou uma aversão à isso ao ponto de que não se documenta nada. Isso, gera um risco que é não existe um registro projetual de uma tomada de decisão; isso não seria um problema se eu tivesse a mesma equipe aqui para sempre dentro de um mesmo produto; mas isso não é uma realidade nossa, porque pessoas são contratadas, pessoas são demitidas, pessoas trocam de times, times deixam de existir, produtos são congelados, voltam depois de um tempo e o histórico de decisão, por que que as coisas são assim, por que as coisas chegaram nesse ponto, se perde, isso é um conhecimento que fica com as pessoas e existe uma sobrecarga de comunicação quando eu não tenho nenhum documento, então, eu tenho que perguntar o tempo todo "Tá como é que isso aqui vai funcionar?" e isso não te permite nem… pô, não posso nem tirar férias, né, se

[55] "Desenhar antes", lógica empregada no Waterfall

não, a máquina para. Que tipo de documentação que eu comecei a sentir falta? Essa funcionais, que expliquem regras de negócios[56], que expliquem o funcionamento, a lógica de funcionalidades, parâmetros de funcionamento como por quanto tempo, por quantos minutos, e não só para facilitar minha comunicação com desenvolvimento, mas é difícil lembrar de tudo, então, eu mesmo posso criar um documento que eu criei para me ajudar. Eu posso usar esse documento para que eu mesmo possa explicar a área, quando, por exemplo, eu tenho novos estagiários, contar como é que tá o projeto sem a pessoa ter que me perguntar tudo, ela pode ler algumas coisas. Pensando mesmo que "Bom, um dia eu posso não estar aqui", outra pessoa assumir o projeto, ela teria esse histórico. Que tipo de documento eu crio? Tem sempre esse questionamento sobre onde eu crio esse documento, sobre qual ferramenta e ambiente em que eu vou acessar isso e eu acabei adotando o Sketch, que é a ferramenta em que eu desenho as interfaces, mais por uma familiaridade e por proximidade ao objeto que será documentado. Eu adotei o Sketch como ferramenta documentadora e outra vantagem de se fazer isso é que eu posso distribuir tanto para designers que têm o software e acesso rápido a esses arquivos, então, se alguém precisar de algo que eu criei, ela consegue fazer isso rápido, quanto para distribuir para desenvolvimento pela ferramenta Zeplin; já que eles não têm o Sketch, eles têm o Zeplin que é o visualizador desses arquivos. Então, eu consigo de uma maneira mais fácil atualizar esses arquivos, toda vez que eu atualizo a interface, eu atualizo a documentação de como ela funciona. Eu faço algumas coisas para aumentar a longevidade deles, uma é: ao ilustrar imagens na interface, eu tento botar imagens atemporais e não mais vinculadas ao conteúdo atreladas a esse momento. Tem uma vantagem em usar imagens "do agora" que é, com o tempo, eu consigo usar as imagens como documento de versão; porém, às vezes essas imagens poderiam causar conflitos dependendo do cliente, então, pode valer usar imagens neutras… Às vezes é difícil garantir que tudo está atualizado, por isso, toda a oportunidade que eu tenho em que eu encontro um documento desatualizado, eu o atualizo, mas eu também não criei uma grande demanda em que eu necessito atualizar; eu deixo até se tornar uma necessidade. Eu não vejo como um problema, é natural: Agora que eu estou revisitando isso, isso se tornou uma necessidade, é o

[56] Por exemplo, como deveria ser o surgimento de publicidade dentro dos vídeos

momento certo disso.

E: O que é documentação na sua opinião?
UX 1: Acho que é um registro das tomadas de decisão de um projeto. As informações que eu tive, compilei, organizei para aquele contexto para aquele problema que eu precisava saber; e eu vou organizar isso de tal forma descritiva para que outra pessoa possa saber, possa se familiarizar com aquele artefato para que ela possa saber quais foram as informações que eu tinha, qual foi o caminho tomado para resolver aquele problema, com quais regras foram criadas para o funcionamento desse pedaço de software. É o registro do momento de um designer. Eu tinha essas informações e eu projetei isso naquele momento.

E: Como é possível lidar com o faseamento da versão ideal de um projeto quando algo "não é possível"?
UX 1: Tem duas interpretações que eu pego para esse "Não é possível": Não é possível *at all*[57] ou não é possível agora? Quando não é possível em qualquer cenário, não é possível fazer isso aqui; cara, vamos encarar essa impossibilidade. Eu acho que se deveria registrar essa impossibilidade, do tipo "Não é possível fazer isso, por isso, fomos por esse outro caminho", daí, temos que atualizar o documento para ele ir para outro caminho. Quando não é possível agora, isso é, faz parte da nossa visão de design, mas não faz parte do nosso faseamento de desenvolvimento, eu acho que a documentação tem que se manter atemporal. Eu gosto de ter uma visão completa sobre a funcionalidade completa em toda sua abrangência. Sobre as entregas, eu não sei… É muito vinculado ao tempo, a cada *sprint*, a cada *release*... Eu gosto de criar documentos atemporais, como se algo precisasse ser criado, documentado e colocado na gaveta para ser feito daqui a um ano; eu acho importante estar documentado. Agora, o que vai ser uma *V1, V2, V3*[58]*,* eu não vejo como algo que precise estar documentado.

[57] Como um todo
[58] Significando nesse contexto: versão 1, versão 2 e versão 3

O que eu faço é, dentro desse arquivo, a parte descritiva é do todo, e a parte projetual, eu tenho um *board*[59] *V1, V2, V3* até chegar na visão do todo e a documentação, dependendo da abordagem do seu texto, ela pode ser neutra o bastante para tu entender essa progressão: A visão está clara e eu consigo entender como ela se reflete no primeiro, segundo, terceiro momento, não é que ela é só válida com a última versão completa. Dependendo da abordagem do teu texto, eu acho que tu consegue dar essa sensação de andamento para a ideia.

E: Deveria existir um histórico?
UX 1: Só se fosse no sentido do andamento progressivo de uma ideia. O lance é que às vezes historicamente, eu tenho um caminho que foi alterado. Eu não manteria na documentação um histórico do que foi abandonado, eu teria a documentação do progresso de uma abordagem e não de diferentes abordagens. Eu não acho que precise ter tudo o que foi feito para um e tudo que foi feito para outro no passado. O que eu documentaria do início do projeto, antes de tomar decisões, seria o que a gente teve que fazer escolhas; por que a gente vai ter a abordagem A e B e vamos seguir a abordagem A. Isso é algo útil para quem chega no projeto completamente fora de contexto e para entender por que o projeto tomou esse caminho, essa direção, mas não que precise de um legado... Eu não tenho problema em arquivar coisas e revisitá-las. Eu gosto de algo meio que… biblioteconomista, que é tipo eu gosto de organizar os arquivos e deixar esses arquivos minimamente "encontráveis" para quando precisar, mesmo que eles não estejam dentro do formato de documentação que eu adoto hoje, que eu consiga encontrar as coisas que eu preciso. É meio que dar dignidade para os arquivos para que eles possam envelhecer com dignidade, então, vão ter arquivos usando outros softwares que a gente não utiliza mais, outros *templates*[60] que a gente não utiliza mais, mas não tem problema, eles registram o momento e eu coloco eles na pasta adequada para o dia em que eu precisar, eu consulto, mas eu não vou ficar dando manutenção em legado.

[59] Quadro, canvas
[60] Página pré-configurada, como um "molde" de layout onde apenas se injeta o conteúdo

E: Você acha que essa maneira de trabalho adotada pela sua empresa é contraditória ao *mindset* de design?
UX 1: Não diria contrário, eu diria que ele nos tira da zona de conforto, porque a gente se colocou talvez com um histórico em uma abordagem chamada *Big Design Up Front* que é *Waterfall*, onde eu tinha um espaço então para projetar tudo, me cercar de uma paz, de um sossego de que "Pô, pensei em tudo, tô entregando esse material aqui sem deixar furos." E depois a etapa inteira era de desenvolvimento. Mas a ilusão dessa abordagem é que essa paz não é totalmente real, porque ao se desenvolver, surgem cenários que eram difíceis de serem previstos enquanto eu estava desenvolvendo [no sentido de projetando, fazendo o design] "O que vai acontecer quando essa situação ocorrer?" e, nossa, eu nem imaginei que essa situação fosse possível, não tenho uma abordagem para isso ainda. Então, não acho que seja contraditório... é uma outra abordagem, nos tira da zona de conforto, só que era um conforto ilusório... eu acho que ele veio a somar para o design. Eu acho difícil fazermos produtos digitais no momento em que estamos agora, que é super acelerado e de ter que se cercar de métricas e certezas sobre o sucesso e a efetividade das coisas... mais do que só um qualitativo abstrato, um "subjetivo" em cima da minha tomada de decisão para uma coisa bem mais objetiva e orientada aos objetivos de negócios. Eu acho que ele só veio a somar. Se as pessoas acham que isso é um desconforto… Eu acho difícil se fazer certo de outra maneira. Eu acho que tem alguns cenários a que *Scrum* talvez não se aplique, mas eu não acredito que seja aqui com a gente. Seriam coisas mais críticas do tipo… se eu estivesse fazendo *software* para controle de reator nuclear, software para controle de sinais vitais de algum paciente. Sabe, eu acho que, realmente, não dá para ficar botando uma versão mínima do produto no ar e testar para ver se funciona, é muito arriscado… Software para controle aéreo de aeroporto… isso é um outro nível. Eu acho requer até um protocolo de engenharia bem mais robusto do que a gente tem de liberdade dentro do desenvolvimento ágil que é de experimentação mesmo.

E: Entendi… Gostaria de fazer mais algum comentário?
UX 1: Não, não, era só isso.

UX2

E: Poderia me dizer com o que você trabalha hoje e há quanto tempo você exerce essa função?

UX 2: Eu trabalho… vish… eu trabalho com design ou comunicação visual desde 2001… e *UX* Design apareceu na minha vida em 2011, quando eu comecei a trabalhar sob a alcunha de *UX* designer… Isso, desde 97, eu trabalho com design de produto e desde de 2000, eu trabalho com comunicação visual.

E: Quais são suas responsabilidades e etapas de trabalho?

UX 2: Hoje, estamos muito reativos, estamos em uma condição de trabalho que é um pouco diferente, mas, assim, eu acho que não foge muito disso. Primeira coisa é o entendimento do problema; a segunda coisa é uma geração de hipóteses após esse entendimento do problema; depois, é a seleção, uma escolha tentando sempre colocar um dado, ou *quanti* ou *quali*, para poder escolher a opção, a hipótese; e aí, colocar *isso pra prod*[61]. e fazer o monitoramento para incrementar e melhorar aquilo que a gente colocou. Por mais que a gente esteja hoje em um cenário de guerra, a gente ainda tá dentro desse *frameworkzinho*.

E: Onde que a Arquitetura da Informação se aplica dentro do campo de Experiência do Usuário? Onde que Arquitetura da Informação aparece no seu trabalho?

UX 2: Então, eu acho que aqui a gente tá tentando mitigar isso… Eu acho que não existe essa divisão entre arquitetura da informação e *visual design*. Eu acho que existem pessoas que têm um *skill* [62]um pouco mais para um lado e pessoas que têm um *skill* um pouco mais para o outro. Quem é "arquiteto da informação" geralmente é um profissional que tem uma visão mais analítica da coisa toda… Ele precisa de um pouco mais de dados para poder criar e elaborar aquilo que ele tá desenvolvendo; não que um cara de visual não faça isso, ele [o arquiteto] só pode ter um pouco mais de facilidade para isso e os *skills* que são mais de estética que acabam sendo aonde o cara tem que se desenvolver mais; algo que não é a natureza desse cara, mas nada que ele não aprenda. Isso é inversamente proporcional para o cara de visual.

[61] Subir o código para produção, no ar para o público
[62] Habilidade

Geralmente, ele é um cara que tem uma estética mais refinada, mais fluida, é mais fácil para ele fazer aquilo; mas ele também pode desenvolver toda essa questão analítica e é isso que a gente tem buscado aqui dentro; tanto por a gente *duplar*,[63] é para misturar e um acabar aprendendo com o outro. Mas esse cara de visão mais analítica, eu vejo ele mais ligado à uma questão estrutural do design mesmo, na busca pelo entendimento do problema para ele poder criar os alicerces da hipótese que vai ser desenvolvida e, uma vez que você já tenha a hipótese e tal, eu não vejo ele tanto na parte de desenvolvimento estético daquilo. Eu vejo ele um pouco mais na parte técnica de documentação do artefato e depois uma análise, uma análise crítica do resultado que aquilo tá impactando, se aquilo realmente está respondendo aos problemas que foram levantados lá na começo… Mas, assim, eu acho que não precisa ter duas pessoas para fazer design. Eu acho que é legal, é bom ter duas pessoas fazendo design, até para poder discutir as ideias e se essas pessoas tiverem *skills* um pouco diferentes, é legal, porque uma vai olhar o problema de um jeito e outra vai olhar de outro e essas coisas se misturam na hora que você está discutindo. Já trabalhei com uns caras mosca branca[64] que faziam muito bem essas duas coisas, mas foram poucos comparados com o tempo em que eu trabalho.

E: O que seria essa documentação técnica?
UX 2: Veja, não é o arquiteto que faz… Eu acho que é mais fácil para ele "arquitetar", digamos assim, os fluxos, a lógica daquilo que tá sendo desenvolvido. Quando eu falo documentação, eu to falando disso sair da mão do designer e isso ir para um outro time de desenvolvimento. Já fiz e já trabalhei com designers que têm esse *skill* de *coder*[65], então, muitas vezes, a documentação era o próprio código; não existe uma documentação PDF, onde você tem documentado o espaçamento, clica aqui vai pra cá, clica aqui vai pra lá.

[63] Trabalhar em dupla
[64] Gíria para profissionais de qualidade excepcional
[65] Quem faz código

Esses designers que têm esse *skill* mais de código, fazem direito no código e meio que entrega já pronto para o desenvolvimento colar aquele *front*[66] aonde quer que seja... mas a relação de *back-end*[67] que tá rolando lá atrás, toda a lógica que tá rolando lá atrás, o design tem que conhecer, ele tem que saber o que que é. Então, quando eu falo em documentação técnica, eu tô falando tanto desse *front* quanto desse *back*. De saber que, pô, por exemplo, se eu tenho essa requisição[68] aqui para o sistema e que o sistema pode me dar duas respostas para esse usuário... esse arquiteto, esse designer tem que saber disso, ele tem que documentar isso de alguma forma. O designer tem que ter a visão analítica da coisa para poder gerar essas opções de tipografia, espaçamento e tal até para poder gerar essas opções do sistema para o usuário ter uma boa experiência de uso do artefato.

E: Você poderia me falar sobre o ciclo de vida desses documentos?
UX 2: Eu acho que o ciclo de vida não é da documentação. É o ciclo de vida do projeto mesmo, porque ao entender o problema, você pode ter 200 milhões de tipos de documentos diferentes para você demonstrar o problema. Na hipótese, você também tem 1 milhão de outras maneiras diferentes para se resolver aquele problema… Na hora que você selecionar a sua hipótese e for solucionar aquilo, você pode documentar isso de várias formas, de maneira que elas não fiquem perdidas no tempo, mas que elas sejam atualizadas e estejam vivas, que é o que o código faz ou é o que a atualização constante dos arquivos fonte do desenho faz também. Se você não faz uma dessas coisas, você vai entrar pelos canos, porque você não vai estar mais falando a língua do teu desenvolvedor. Ele vai estar falando uma coisa e você outra. Eu acho que é meio que um…para você trabalhar com a metodologia ágil, o cara tem que saber o que é uma metodologia ágil e tem que ter a atitude ágil. Eu acho que não é nem o *mindset* nem o entendimento, é a atitude *agile* mesmo do cara chegar e falar "É só um botão? É só um botão. Então, porra, beleza, eu vou sentar ai do seu lado e a gente só arrumar o botão" e vai. Eu não vou criar uma

[66] Porção do desenvolvimento voltada para construção da interface com o usuário final, também chamado *front end*
[67] Porção do desenvolvimento voltada para desenvolvimento das aplicações web, também chamado *back*
[68] Condição de funcionamento e execução de um sistema

pasta, criar um arquivo, desenhar só esse botão, salvar no servidor e depois eu não sei mais onde tá, onde foi que eu mudei.Prefiro virar para o cara do desenvolvimento e falar "Velho, é assim, sobe". Subiu. O que *tá em prod*, é isso a verdade.

E: Você sente que algumas informações de arquitetura faltam na documentação ou que são mais difíceis de serem documentadas?
UX 2: Eu acho que, na verdade, é muito mais uma questão de tipo de projeto. Uma coisa é você fazer um site em que ele é extremamente passivo, você desenvolver uma coisa visual que é extremamente passiva, onde o máximo de interação complexa é o contato. O projeto que vai definir o tipo de entrega que se deve ter e, muitas vezes, o tempo não te dá tempo de pensar que tipo de entrega você vai fazer, que é o que tá acontecendo com a gente, a gente tá aprendendo a fazer enquanto a gente tá fazendo. O contexto em que a gente tá inserido para trabalhar, ele faz parte do problema, não tem como tirar isso. O tipo de documentação que você vai entregar, tá diretamente relacionado ao tipo de problema que você vai resolver e, quando eu to falando de resolver, não quero dizer você só elaborar a hipótese, eu to falando do artefato pronto lá no final. Você tem que ter uma documentação que explique o que que aquilo ali é, o problema tem que estar claro para todo mundo e aquela documentação tem que explicar de que maneira você está resolvendo aquele problema.

E: Qual o valor em se produzir esses documentos?
UX 2: Eu acho que a documentação, ela é um canal de comunicação, onde você entendeu o problema, gerou suas hipóteses, selecionou sua hipótese e você tá passando aquilo para alguém desenvolver, para alguém incrementar aquilo que você está documentando. É mais uma ferramenta de comunicação, na verdade. Poderia ser, inclusive, o próprio artefato, onde o código é o próprio documento. O designer poderia saber fazer o HTML/CSS, o JavaScript em caso de micro-interação, e entregar isso para desenvolvimento "plugar" essa coisa no *back*, mas aqui dentro, a gente não faz isso. Às vezes, você não precisa nem desenhar quando você tá em *Agile*; você senta do lado do desenvolver e vocês dois juntos discutem e acham um melhor caminho e aquilo se desenvolve dentro do código.

E: Mas cadê a referência para que outras pessoas possam olhar para aquilo e entender?
UX 2: Não precisa, é o código. Do tipo, pô, arrumou aqui, foi para *prod*, lá é a documentação. É uma maneira arriscada, mas é o que o *Agile* prega. Que *tá em prod* é o que é a documentação. *Agile* é rápido, então, muitas vezes você não precisa criar o arquivo para entregar para o seu desenvolvedor e, uma vez que ele colocou aquilo no código, esse arquivo estático no tempo, ele vai morrer, ele não tem porque existir mais, uma vez que você pode ir lá em produção, ver o que é que tem e inteirar em cima daquilo.

E: Você acha que o processo de ideação deve fazer parte da documentação?
UX 2: Eu não acho que a ideação seja a documentação. Eu acho que na hora que tu tá entendendo o problema, fazendo tuas pesquisas para entender o problema, entrevistando as pessoas, teus *stakeholders* para entender o problema real como ele é, muitas vezes, tu vai acabar rafeando[69], rascunhando coisas para tentar chegar a um entendimento, aquilo ali é um documento, é um rascunho. A partir do momento em que isso se torna o código, se torna o artefato em si, aquela tua foto da parede perdeu o sentido de existir, porque tu já tá com aquilo pronto, já tá com aquilo documentado.
Outra coisa é que se o código se torna a documentação, para ele [o desenvolvedor] aquilo ali também vai bastar, porque, ele não vai criar a mesma tela várias vezes, porque ele vai saber que aquilo é um conteúdo dinâmico, porque ele entende o código, ele entende o HTML, ele entende estrutura de *back* sobre que ele tá projetando em cima.

E: Mas isso não é muito… não-empático? Imagina quando uma pessoa sai do time, quando chega uma nova, quando alguém sai de férias...

[69] Significando rascunhar

UX 2: Então, esse cara parte do que *tá em prod*. É muito melhor do que o cara ter que ficar olhando para um monte de tela desatualizada. Porque muitas vezes o que acontece é que o projeto é muito grande, muito dinâmico, muito rápido e você vai lá, desenha uma tela, teu desenvolvedor desenvolve, põe ela em produção. É muito mais rápido e barato, você alterar um texto só lá em produção do que você pedir para o designer ir lá, atualizar a tela, salvar a tela, entregar para o desenvolvedor ir lá alterar. Então, se o cara chegou e é novo… "Pô, navega ai, cara. Eu tenho aqui um fluxo de cenários e esses são os cenário que eu atendo; e dentro de cada cenário, eu tenho essas *features* aqui, o conteúdo delas", isso nada mais é do que um "Word' [um documento escrito no geral]. Não precisa ter o artefato inteiro desenhado, documentado e arquivado para que aquele cara entenda e, *em prod*, consiga ficar atualizado com o que tá desenhado, implementado de verdade; a partir daí, ele consegue desenhar o resto. Veja, não é uma negação da documentação isso, isso é mais pelo pensamento do processo *Agile* mesmo. A documentação em produção, ela funciona, mas o designer tem que entender o que ele tá lendo ali, não adianta nada ele não entender.

E: O meu receio é que o *Agile* para mim parece ser muito uma oportunidade de pensar muito micro e não pensar no todo...
UX 2: Acho que mais uma vez depende do projeto em que você tá trabalhando. Não é nem a metodologia do projeto, é mais o *mindset* do cara em relação ao contexto em que ele tá, entendeu? Se você não entender o contexto em que você está inserido e querer ficar ideando muito, o projeto vai te engolir e você vai virar gargalo no projeto. Você tem que saber a hora em que você tem que sair, tirar o pé, fazer a próxima entrega e partir para a próxima coisa, que é difícil às vezes. Às vezes, a gente quer entregar a coisa mais perfeita do mundo e não dá. Você tem que tirar o pé, entregar o que tem e partir para a próxima e num outro momento você vai conseguir revisitar aquilo para conseguir ajustar.

E: Considerando o material entregue por UX e o que foi desenvolvido, o que você diria que é a documentação final: o material de UX ou o código?
UX 2: Código é o final mesmo.

E: A organização em que você trabalha adota algum tipo de metodologia de trabalho? O que essa metodologia tem de bom e o que ela tem de ruim?
UX 2: O que ela traz de bom pro design… Faz o designer pensar mais "ágil". É difícil responder… porque, ainda mais que se fala das grandes ferramentas de UX, de você fazer grandes pesquisas, depois você documenta, depois você valida o que foi desenvolvido e tal… Vai ter uma porrada de brecha. Mas por outro lado, você ter todo o tempo para projetar antes, ajuda a trazer uma visão de futuro para todo mundo que está envolvido no projeto, para que todo mundo olhe para aquilo e fale "Ah, é lá onde a gente quer chegar"... É difícil responder essa pergunta também, porque o contrário também é válido: Quais as vantagens de se ter um designer em um modelo *Agile*? Porque, no final das contas, se o designer não tem o tempo de parar e idear as coisas para projetar, depois as coisas ficam complicadas… Eu acho que tinha que ter uma mistura tanto do *Agile* quanto um pouquinho de *Waterfall* dando o momento do projeto. Eu acho que se você tem uma demanda ou um problema claro, eu acho que o designer pode partir antes e tentar entender o problema por pesquisas, conversas com os *stakeholders* e tal e, uma vez ele delimitando esse problema, o *Agile* funciona muito bem, porque ele só vai desenvolvendo e interando. Ele não tem entregas gigantescas com aplicações N da mesma coisa mudando só um elemento. Então, eu acho que para o designer, o *Agile* é bom, porque ele aprende a ser ágil e pro *Agile*, é bom ter um designer, porque ele tem uma visão mais clara do usuário que ele tá tentando atender e menos pragmática, assim, do desenvolvimento por si só; então, é difícil, porque os dois ganham, mas os dois perdem ao mesmo tempo, mas eu acho que é possível achar um equilíbrio em que o designer consegue aproveitar o *Agile* para não perder tempo fazendo documentação e aproveitar esse tempo para fazer ideação. Então, no final, eu acho melhor tu ter um time *Agile* com um designer que saiba trabalhar bem com *Agile* para ele não ficar perdendo tempo em documentar hipóteses e ideias; com um *raff*[70] simples, ele consegue fazer essa comunicação e ganha tempo para ficar pensando no futuro e ficar evangelizando as pessoas com relação à essa visão de futuro.

[70] Rascunho

É tipo, não adianta tu ter uma ideia do caralho se as pessoas não vão entender aquilo que você querendo desenvolver... Para o designer é muito bom, porque você interage com o seu protótipo. Se você sabe código, você pode fazer isso sozinho, se você não sabe, não tem problema, você pode conversar com o seu desenvolvedor.

E: Você acha que o *Agile* pensa nesse futuro?
UX 2: Acho que pensa. Eu acho que até o ponto em que você consegue fazer com que as pessoas que estejam contigo consigam entender qual é a visão de futuro, você pode ter isso documentado ou isso dentro de uma reunião muito fácil. Basicamente, fazer as pessoas comprarem a sua ideia.

E: Quais são as maiores dificuldades de lidar com esses documentos dentro desta metodologia?
UX 2: Eu acho que é falta de tempo para você projetar o futuro, eu acho que essa é maior perda que design tem no *Agile*. E no, caso de sistemas complexos como a gente vive agora, é de depender de um terceiro para simular um certo fluxo, um certo cenário para poder ter insumo suficiente pra desenvolver.

E: Você acha que essa maneira de trabalho adotada pela sua empresa é contraditória ao *mindset* de design?
UX 2: Que tem conflito… Pois é, era isso que eu tava te falando, eu acho que o design pode ser meio burocrático pro *Agile*, mas porque ele precisa ser burocrático, e, às vezes, o *Agile* é muito rápido para o designer sendo que ele precisa ser um pouco mais burocrático para ele pesquisar o problema e entender o problema.

E: Gostaria de fazer algum comentário, observação ou sugestão?

UX 2: Acho que vale ficar claro que quando você está trabalhando com metodologias de projeto, essas metodologias têm que estar adequadas ao problema que você quer resolver. Eu acho que não é a metodologia que vai definir o problema a ser resolvido, é o problema que vai definir qual é a tecnologia, qual é o modelo de trabalhar, e o problema é o contexto inclusive, não só o o que o artefato precisa resolver, mas o contexto inclusive. Então, mais do que ser um carimbo de Linkedin, é esse entendimento sobre quais os tipos de problema que eu posso vir a ter, essa elaboração de cenários, para eu não me quebrar lá na frente. Para eu ficar com a minha entrega sempre, pelo menos, razoável. Proximidade do design com relação a outras áreas de projeto para além do desenvolvimento. Que tem que conversar, tem que. Que o designer é o animador desse processo, é um fato. O problema é até onde o designer pode ir dentro da estrutura da empresa em que ele tá. Às vezes, dependendo do tamanho da empresa e da maneira como a empresa vê o design, você teria a possibilidade de alterar o produto, mas isso é muito relativo ao contexto. Eu acho que falta um pouco essa coisa de contexto nas suas perguntas, assim. E eu acho que o design não pode se contentar com a maneira como as coisas são feitas por si só, o design tem que olhar para trás e falar "Pera, é o problema maior de eu só fazer isso que eu to fazendo, eu posso ser mais proativo", mas isso depende muito do cenário em que ele tá inserido… Então, essa visão holística do design relacionada a contexto, com relação a problema, com relação a usuário, com relação a… é um trabalho de mediação efetivamente. Você tá mediando todos os envolvidos naquele projeto, seja ilustrando através de tela, seja codando com o cara do seu lado. Tem todo esse contexto que vai te ajudar ou vai te estrepar na hora do uso de um *Agile* ou do uso de um *Waterfall* qualquer… Mais ou menos isso... A seleção do tipo de *framework* que você vai usar está muito atrelada com contexto e ao tipo de projeto e a documentação acaba sendo uma consequência disso tudo.

E: Designer precisa saber codar?

UX 2: Acho que para você ser designer, você tem que gostar de aprender. Eu acho que o design faz parte da essência do ser humano: designar forma, designar nome, designar coisa, designar função a coisas. E eu acho que se você está aberto a aprender coisas, eu não acho que você tem que saber codar, mas se você vai fazer uma entrega, não custa nada você entender qual é a língua desse cara para você conseguir falar. Desenho enquanto representação e ferramenta que tangibiliza uma ideia. O desenho é uma ferramenta de entendimento. O desenho em si não precisa ser esteticamente perfeito, ele pode ser um raff [rascunho], pode ser um rabisco, o importante é que quem olhe consiga entender o que está ali. Se você vai fazer uma entrega daquilo, dependendo do tipo de projeto, você vai precisar de uma estética mais rebuscada, uma estética mais suja. (...) O designer, ele é um resolvedor de problemas, um facilitador das coisas que usa o desenho como uma ferramenta de comunicação e estética é uma consequência disso tudo. O conceito de stakeholder é bem abrangente. Ele pode ir desde o demandante passando pelo cara que e o fornecedor até o cara que consome no final e o designer é transversal a tudo isso. Ele e o animador desse processo, ele que faz a comunicação entre todos esses envolvidos, entre todos esses stakeholders. E um assessor de stakeholders. Designers digitais são projetistas de informação: Como esse fluxo de informação vai acontecer entre o sistema e o usuário. Designer precisa estar a um passo adiante. Eu acho que o designer tem que estar mesmo. Então, repertório, tudo o que envolve a gente na nossa sociedade… Quanto mais repertório mais a um passo à frente ele vai estar. A falta da gente ter um desenvolvedor ali do nosso lado, a falta de uma cultura *Agile*, faz com que a nossa documentação fique desatualizada e a falta de comunicação diária, de um *daily*, de um planning faz com que a gente tenha que ficar constantemente manutenindo a tela para que mudanças sejam registradas para que sejam implementadas. Enquanto você está criando as suas hipóteses, tu tem que entender o seu sistema. Não é você começar a desenvolver junto com o desenvolvedor, é você ter um tempo mesmo de pensar, de entender o problema, gerar as hipóteses, selecionar a hipótese e fazer uma entrega… Não, teu projeto não acabou ai, tu não resolveu o problema de uma vez só, por isso, você tem o protótipo.

Tudo o que eu desenhei antes [antes daquilo que efetivamente funciona] não é documentação, é rascunho. Documentação de verdade é aquilo que funciona, que dá certo. Quando a gente não tem o desenvolvedor junto da gente, aquilo vira uma documentação e ai, depois, temos que lidar com débitos de desenvolvimento [do que era um plano que não deu certo e, ao invés de rever o plano, remendamos ele]
O que você fez até chegar nessa solução (pesquisa, bench, rascunho, etc), na hora que o cara plugou isso para funcionar, podem e vão aparecer problemas. É nesse momento que você como designer tem que inteirar com o seu desenvolvedor sem precisar de documentos, arquivos e tal; o que está no ar, se torna a sua documentação. (...) No máximo, você vai ter uma biblioteca com elementos para poder gerar esses cenários e tal. (...) Até você chegar em produção ou ter um protótipo funcional daquilo, tudo o que você fez é rascunho. Isso [o que "real"] é a tua documentação. É isso que o *Agile* vê e é isso que o design tem que aprender do *Agile*; assim como o *Agile* tem que aprender que, se você não projetar antes, só sair fazendo a documentação, ou seja, só sair codando e *colocando em prod.*, também não vai funcionar. (...) Essa é a função do design, dar essa visão de futuro da melhor maneira possível, da maneira mais completa. De repente, enquanto você estiver prototipando aqui, você vai ver que tem um furo gigante e tem que voltar para prancheta, tem que arrumar tudo de novo, não só um pedacinho. Eu posso fazer um template para um cenário, então, toda vez que aquela coisa acontecer, é esse template a ser usado e eu não preciso desenhar todas as telas aplicadas.
O designer tem que estar na frente para criar a visão de futuro e quando essa coisa estiver sendo desenvolvida, tem que entrar junto do desenvolver e interar e melhorar aquilo e depois quando aquilo entrar em uma larga escala ou for altamente personalizado… Você estando ali do lado de quem faz as coisas, é muito mais fácil aquilo se tornar uma documentação mesmo, mas tudo é o contexto, começa ai o seu problema de design.

APÊNDICE F - Respostas das entrevistas semi-estruturadas para desenvolvedores

Dev 1

E: Eh… Então, eu to fazendo um TCC que é sobre documentação de UX dentro de empresas que adotam o modelo ágil de trabalho. Porque eu percebi que algumas coisas acabam saindo do controle, algumas documentações não estão atualizadas… Daí, eu queria saber qual que é o ponto de vista de desenvolvimento sobre isso e como é que UX poderia trabalhar melhor para ser mais rápido, fazer documentações mais consistentes.

Dev 1: Eu vou te dizer que… a visão que eu tenho de documentação de UX como documentação no geral, que pode ser uma documentação de arquitetura[71], por exemplo, que caímos nos mesmos desafios que é muitas vezes a gente quer pensar no todo e fica horas pensando naquilo, faz grandes planejamentos, define as coisas e depois vai embora. E daí tu cria um problema de comunicação entre quem fez essa documentação e quem vai executar depois, porque como qualquer plano, é na hora que tu bota na rua que tu descobre se aquilo fica de pé ou não ... e a pessoa quando fez, ela botou toda energia, carinho e atenção possível; quanto mais ela gastou energia fazendo, menos, eu diria que é inversamente proporcional, a abertura que ela vai ter para mudar as coisas. Então, acaba acontecendo que o time acaba dando uns feedbacks como "Ah, isso aqui não dá para fazer ou se fizer, vamos ter esse tipo de compromisso e tal" e as conversas tendem a ser bem conflituosas ao invés de serem convergentes justamente porque tem esses distanciamentos. Então, tô pra te dizer que, das minhas experiências com UX foram com aquelas [experiência] a gente tinha um profissional pensando em UX, fosse o cargo que a pessoa tivesse como UX, PO, Analista de negócios, com o resto do time todo o tempo. E eu não estou advogando que tudo tem que ser feito à medida em que vamos desenvolvendo, não. Pensar a longo prazo é importante. Tem um monte de cerimônias como ideações que ajudam a ter uma ideia de para onde é que tu tá indo, né, mas nada sobrevive ao crivo, um, do desenvolvimento, que é quando tu tangibiliza as coisas, mas, principalmente ao crivo de quem vai usar isso depois.

[71] 05:52: Quando falo "arquitetura", digo, arquitetura de software.

E daí, o que acontece muitas vezes, pelo time de desenvolvimento estar desenvolvendo de maneira interativa e incremental, bota na rua as coisas, coleta feedback e tem que voltar para a figura de UX que está distante e dizer "Cara, isso aqui mudou" e daí [a figura de UX responde] "Não, mas eu pensei assim. O meu design está perfeito", porque a pessoa conhece bem tecnicamente, mas às vezes, o que a gente conhece não é o suficiente. É o crivo do usuário que vai fazer a gente conhecer mais, né. Então, por isso, esse ciclo é importante. Por isso que eu acho que com o profissional "tando junto", ele enxerga esse ciclo fechando e daí se torna mais aberto, mais receptivo; e daí, com isso também, a documentação, ela é verdadeira, digamos assim. Uma coisa que eu brinco é que, muitas vezes, a gente usa uma ferramenta seja ela qual for, pode ser uma Wiki… A Wiki é o lugar onde os documentos vão para morrer, porque tem uma pessoa atualizando, mas ninguém nunca lê aquilo."

E: Sobre esse lance né, do pensamento do designer que ele tem que ter o todo em mente, isso é uma coisa que… Eu queria conduzir essas entrevistas tanto com designers quanto com desenvolvedor, porque eu acho que talvez o mindset de design seja meio diferente, assim… Eu acho que eu tenho dificuldade ainda de pensar num todo e, sei lá, às vezes o que vocês precisam, às vezes o que é fazível, o que é necessário, primeiro é um step desse tamanho [dando a entender que é pequeno] e a gente já pensa no todo. Sei lá, é uma dificuldade particular que eu tenho, mas eu também não tenho tanta experiência assim… maneiro o seu ponto.
Dev 1: Eu diria que essa sua visão, ela é compartilhada por algumas pessoas, sabe? E é a mesma…. o mesmo ponto… a mesma forma de trabalhar, eu diria que pessoas com o papel, com o cargo de arquiteto de software também tem. Tem que pensar em toda a solução, de como é que vão ser todos os componentes, em como que eles vão se inteirar; só, na verdade, isso tinha que acontecer de uma maneira mais orgânica. Sim, é importante tu saber qual é o norte, para onde tu tá indo, mas tem coisas que tu só vai descobrir a medida em que tu, um, bote de novo a mão na massa, né, e, dois, passe pelo crivo de quem vai usar isso no final.

E: Uma coisa que você comentou que eu achei interessante e que eu queria comentar é que relacionada à documentação de arquitetura… Foi isso?
Dev1: Isso. Quando eu falo arquitetura, eu quero dizer arquitetura de software, não arquitetura como engenharia civil.
E: Não, não, é que eu pensei em arquitetura, tipo, arquitetura realmente da informação.
Dev 1: Ah, não, não, desculpa. Arquitetura de software que eu tava falando. Toda vez que eu falar em arquitetura quando tu for revisitar esse material, eu to falando de arquitetura de software, não é arquitetura de informação. Foi mal. Bom ponto para deixar claro isso.

E: Eu fiz aqui um roteirinho, eu vou te perguntar algumas coisas dentro dele, tá? Há quanto tempo você trabalha com desenvolvimento.
Dev 1: Eu vou te dizer que, oficialmente, com desenvolvimento de software, desde 2006.

E: Quais são suas responsabilidades e etapas de trabalho dentro da sua função?
Dev 1: Tá, hoje, o meu papel, oficial, digamos assim, o termo que a empresa usa, é GA, que é gestor de área. Na prática, eu tenho duas grandes responsabilidades. Uma delas é trabalhar com questões relacionadas à gestão de pessoas, outra delas é trabalhar com questões relacionadas a desenvolvimento de software, do ponto de vista de agilidade.
Então, tem vários lugares, vários times daqui que usam o termo Scrum Master, por exemplo, como sinônimo disso…. Mas se você pegar o Scrum Guide, vai falar só de agilidade, mas aqui, tem toda essa questão de carreira das pessoas, acompanhamento, descobrir que que elas tão afim de fazer no próximo ano, quais são os eventos, aumento de salário, todas essas coisas 'cai' para a gente também.

E: Qual você diria que é o papel do designer dentro do seu *workflow*? Como que design se integra a isso que vocês fazem?

Dev 1: Então, para mim, ele tá muito conectado com a minha fala anterior, inicial ali, que é o designer tem que ser um membro ativo na equipe de desenvolvimento, porque ele é o profissional que tem uma formação diferente das pessoas que trabalham com desenvolvimento puro, então, ele vai trazer algumas questões relacionadas à, sei lá, jornada de usuário, que talvez os devs não tenham… então, isso é importante. O designer tende a ser também o cara que vai entender umas métricas para saber se aquela solução é válida e útil, que talvez também os devs não tenham. Então, eu acho que ele tem que ser assim um membro no time. Tem gente que advoga que não há espaço... não há trabalho 100% do tempo para um designer dentro do time. Eu acho que não tem um regra única, depende da situação. Eu acho que tem times que vão demandar mais do profissional, tem time que vai demandar menos, então, tu pode ter de repente, um profissional compartilhado entre times afins, mas, independente disso, um time sem alguém com habilidades de design, é um time órfão.

E: Com relação à documentação, como a gente tava falando antes, quais que você diria que são geralmente os problemas de uma documentação de design quando ela chega para desenvolvimento? Tipo, é falta informação, ela chega desatualizada…?
Dev 1: Eu acho que, uma, uma coisa que é inerente à documentação, que para mim, é sempre um problema, é receber um pedaço de papel, seja ele físico ou eletrônico e não ter um conversa em cima dele. Então, ele sempre cria um problema de comunicação. A segunda coisa é uma desconexão com o que está sendo desenvolvido; muitas vezes acontece, porque, tu pode ser um designer que foi lá, falou com o usuário final, falou com o stakeholder, sei lá com quem for, mas não falou com o time, então, ele pode potencialmente ter criado uma ótima solução, mas que ela não é factível no momento em que o desenvolvedor pega aquilo porque não houve um cuidado com conexão. Isso é um problema que pode acontecer… Tu pode sim estar desatualizado, porque aconteceu a conexão, mas, sei lá, o designer ficou trabalhando naquilo durante três meses e o software desenvolveu para outro lado. Quando essa documentação chega, ela não encaixa, isso pode acontecer também…

Eu acho que essas seriam as coisas que me vem à mente imediatamente. Outros problemas que eu vejo é... mesmo que a informação chegue de maneira adequada, à medida que o software for sendo desenvolvido, se ninguém ficar cuidando dessa documentação; se eu tiver um membro novo time que entre dois, quatro meses depois e tente usar essa documentação pra fazer algum *on boarding*, para ter entendimento, já não serve também, então, tem um problema também de manutenção dessa documentação na medida em que o software vai sendo construído e evoluído.

E: Sobre essa questão da diferença entre documentação e software, o que você diria que seria a documentação final? Porque tipo, esse lance da documentação, ele também a questão das versões, né, primeiro a gente faz esse tanto, depois a gente faz esse outro tanto e cresce isso. O que você diria que seria a documentação?
Dev 1: Para mim, a documentação ideal é aquela que tem um público. Então, se você não tem um público, a documentação não precisa existir. Dito isso, eu "te dou um grande depende", na verdade, que é depende do time, depende da empresa que tu tá desenvolvendo, depende qual é o teu público, então, muitas vezes, tu vai ter uma documentação que ela é para usuário final e uma documentação que é para time de desenvolvimento. Então, por exemplo, às vezes, você tem um aplicativo que quando você entra, ele te dá um passo-a-passo, um highlight, coisas que são para te dar uma ensinada, daqui a pouco, aquilo é muito mais um tipo de documentação importante do que páginas e páginas de um manual. Para mim, aquele tipo de documentação em que o cara entrou no sistema e tem uma nova feature tende a ser muito mais interessante, muito mais relevante do que um manual super completo, mas que ninguém vai ler nunca. Então, não adianta tu ter um pdf todo bonitinho, todo bancana num site qualquer se o número de acessos àquele pdf é zero. Então, dá aí de novo… Depende muito de qual a situação em que tu tá.

E: Mas a documentação que chega para você em desenvolvimento, você geralmente fica satisfeito com ela?
Dev 1: O meu ponto de definição para poder te responder isso seria 'qual é o tipo de conversa' que eu consigo ter contigo quando tu me dá essa documentação: Se as minhas conversas são muito... 'alto nível', no sentido de 'essa cor aqui era melhor do que a outra', tipo, isso é meio que bobo… é meio que supérfluo. No momento em que você me dá a tela, eu entendo o tipo de fluxo e eu entendo que que aquilo vai ajudar na vida da pessoa, tipo, ah, 'ela vai ter menos cliques, ela vai ser mais feliz, ela vai gastar menos tempo', (...) qualquer coisa nesse sentido, de que seja mais humano, ai, eu te diria que aquilo é uma documentação melhor. E daí, de novo, depende de como o time está "azeitado" para ser construído… Às vezes, a gente personifica demais, 'ah, o designer fez um troço ruim', mas, às vezes se o time não está acostumado a trabalhar com designer, pode ser bem desafiador fazer uma documentação em um nível adequado. Eu conheço gente que é… aquele expressão em inglês, *code monkey*, macaquinho de código. Então, que é que tem uma documentação super bem estressada, dizendo qual é o pixel por pixel, onde é que vai, qual é a cor com o hexadecimal definido e eu acho que isso é meio… sei lá. Desenvolvimento de software é uma atividade de criatividade, então, se tu só me mandar fazer alguma coisa, eu acho que isso é uma má documentação também, então, o excesso de informação pode ser ruim também se ele não deixar nenhuma margem para conversar contigo sobre aquilo.

E: Interessante, legal, eu não tinha pensado nesse ponto. Mas uma coisa que eu vejo que você falou bastante, é sobre essa questão da conversa. É você realmente ter o time também fisicamente perto. Sei lá, vocês compartilharem uma baia. Mas para uma galera nova que não conhece ágil ou então, pessoas que trabalham distantes umas das outras, você acha que a documentação ajuda nesse sentido? Para a pessoa tentar entender… É, porque pode ser um processo meio oneroso… Tipo, o balanço entre você ter que ser ágil e documentar.

Dev 1: Documentar as coisas tende a ser a maneira tradicional de fazer… de solucionar os problemas… e o meu pé atrás com isso é porque, ele normalmente vem do medo. Ele vem do, da expressão inglês, essa mais chula, do '*cover your ass*'. Por ter as coisas bem documentadas, eu tenho um álibi: 'Olha, eu fiz a minha parte, quem não fez, foi você'. Então, por isso que, em um ambiente ágil como a gente quer ser, independente se são times daqui ou time de terceiro, eu acho que são conversas cara-a-cara mesmo, seja através de uma *vídeoconfi* vão solucionar melhor os problemas do que dizer, 'ah, tá lá na documentação' e pronto. Uma coisa é tu ter uma documentação, porque, sei lá, teu time de desenvolvimento está em outro fuso horário, então, você não está sempre disponível para eles, ai, beleza, é importante que eles também consigam trabalhar enquanto você não tá online; agora, sempre que possível, a documentação tem que ser um meio para a conversa e não o fim.

E: Porque a minha ideia era que, ah, essa é a ideia de UX, então, desenvolvimento tem que fazer. Então, a gente faz da maneira mais bonitinha possível, para sei lá, ter...

Dev 1: Isso é super anti-ágil, super anti-ágil, porque isso é bem, bem o modelo cascata. Alguém faz um pedaço e depois passa para o time de desenvolvimento que só executa. Não tem nada com UX, pensa em negócio: Alguém vendeu uma solução e entregou para desenvolvimento e desenvolvimento tem que cumprir… Cara, se o que tu vendeu não serve para quem comprou, tu tem um problema. Então, isso é um problema que a agilidade quer resolver, indiferente se é UX ou vendas ou o que for.

E: Quais são as maiores dificuldades de lidar com UX dentro dessa metodologia? Porque, ainda dentro da minha cabeça, vocês de desenvolvimento ainda precisam ter um referencial de UX que entrega, né?

Dev 1: Sim, com certeza, eu acho que todos os referencias que UX entrega traz para desenvolvimento são excelentes e eu acho que não é por ai que passa o problema. O problema passa pelo distanciamento. Eu acho que quando a gente tem um time que funciona como silo ao invés de profissionais que têm aquela bagagem dentro do time de desenvolvimento, ai tu tem um problema.

E: Então, todo mundo tem que estar na mesma página… Foi mais ou menos isso que você quis dizer?
Dev 1: Sim, exato. Porque não tem problema, você como UX que tá pensando em toda jornada do usuário, em todas as múltiplas jornadas do usuário dentro do seu produto, ter uma visão de futuro e ter uma visão de mais curto prazo; isso é importante, isso é fundamental que você tenha. O problema que eu vejo muito às vezes é: A gente tem em separado UX num pedaço e dev no outro. Então, ux pensa muito no futuro, desmembra tudo, faz documentos lindíssimos, manda para cá, diz "cumpra-se" e vai para a próxima coisa. Então, isso, além de criar esse distanciamento, não cria aquele loop de feedback que eu tava te falando antes, que é, beleza, quando dev entregar para o cliente e isso voltar, isso volta só para dev, não volta para ux. Tu tem que pegar o cara de novo, daqui a, sei lá quanto tempo, para ele ser emprestado para o seu time de novo e ter um novo projeto… E, tipo, anti-ágil total.

E: Como você imagina que isso pudesse ser diferente? Tipo, imagino que isso aconteceria por convivência, tipo, UX tá lá o tempo todo...
Dev 1: Sim, convivência. UX tem que fazer parte do time e tem que olhar para o todo. Você vai para o resto do seu time de UX para pensar qual a grande de visão de UX para todos os produtos? Beleza, lindo, mas, agora, na prática aqui, o que que o meu usuário tem de diferente do usuário do time da Marcela? Sabe, é diferente, então, por mais que a gente tenha que pensar em toda UX para todos os produtos, eu tenho [no meu time] uma peculiaridade e você [seu time] tem outra. O meu UX e o seu UX têm que ter o olho no todo, mas tem que ter o olho no produto em que tá imerso.

E: E nessa comunicação então entre UX e desenvolvimento, quais devem ser pontos de atenção para a entrega de ux enquanto tentamos equilibrar o todo [no sentido de visão de longo prazo] e o micro, mais prático [no sentido daquilo que é mais prático, mais curto prazo]?

Dev 1: Então, a meu ver, eu acho que é um pouco do que a gente falou, mas eu não consigo enxergar um trabalho sendo bem feito, e daí, independe se é ux ou dev, sem as fases clássicas de entender qual é o problema, criar uma solução, aplicar aquela solução e ver qual foi o resultado. Não tem como fugir dessas quatro etapas. Ux tem que entender que, por mais que não seja ele que vai colocar a mão na massa em todas elas, tem que entender que essas etapas estão acontecendo e tá aberto a escutar coisas que ele não tinha escutado antes que talvez façam com que ele tenha que mudar coisas que ele achava que fossem lindas e maravilhosas, porque tudo é lindo e maravilhoso, dentro da nossa cabeça, mas é a realidade que vai deixar aquilo lindo e maravilhoso de verdade. E isso é um problema que, de novo, nem só ux tem, dev também tem.

E: Que tipos de informação você precisa que ux passe?
Dev 1: Eu diria que é importante ter a noção macro da necessidade que a gente tá resolvendo, e aí, a partir dela, entender qual é a solução que está sendo estabelecida, do tipo, qual a jornada do usuário a usar o seu produto e, daí, tu vai esmiuçando: quais são os eventos importantes, quais são as telas importantes, quais são as interações importantes; depois vai esmiuçando de novo: ah, dentro dessa tela, quais são as informações que vão estar disponíveis, quando elas vão estar disponíveis, como é que elas se conectam entre si, aí depende quanto mais longe ou quanto mais perto tu vai estar da implementação. E, por fim, né, como é que que tu mede para saber se as suas hipóteses foram contempladas ou não, eu acho que isso é importante UX fazer também. Eu vejo muitas vezes isso não acontecendo. Eu acho que tu não deveria começar a desenvolver nada sem entender qual o problema que tu quer resolver, qual é a solução macro que tu quer resolver e sem o mínimo esboço de tela; agora, os detalhes dos por menores[72], eu acho que podem ser estabelecidos em tempo de execução.

E: Detalhes e pormenores, tipo o que, por exemplo?

[72] Detalhes de tela. Isso, ele aponta, pode ser resolvido por teste AB, por exemplo.

Dev 1: Ah, sei lá, esse botão fica melhor na esquerda ou na direita? Daqui a pouco, eu vou ter que criar uma tela com ele na esquerda, na direita, criar um teste AB, vou ver o que que sobrevive e defino que é ali ao invés de eu ficar discutindo em 15 reuniões com 80 pessoas diferentes até fazer uma votação para descobrir quem é que gosta mais da esquerda ou da direita. Tipo… azar. Quem tem que gostar é o cliente, não é nenhum de nós aqui nessa sala.

E: Então, tipo… não sei assim. Uma hipótese de solução seria: Você, em algum momento do desenvolvimento, precisaria talvez de alguma ideia mais macro e aí quando for o momento das entregas, uma visão mais detalhada sobre o que se fazer?
Dev 1: Sim, por exemplo, tu… Eu sei que alguns times ai do Rio têm usado o conceito de Concepção de produtos. É um momento de ideação que pode durar um dia ou uma semana, onde as pessoas tentam entender qual o problema que quer ser resolvido e criam uma solução. Nesse tipo de situação, tu tende a ter um planejamento macro, de entender qual é o problema e entender quais são as jornadas de usuário que tu vai criar para resolver este problema. Então… eu acho que isso é muito importante para um *kick-off*, para um gatilho de começo para desenvolvimento. Só que agora, quando cada um dos funcionários for pegar uma funcionalidade para implementar e ela tem alguma coisa mais forte relacionada a, sei lá, arquitetura de informação, eu diria que tem que ter no mínimo, um bom esboço. Talvez não precise ter o detalhe do pixel, mas tu precisa ter uma ideia de referencial. Porque, por essa diferença, ao implementar, tu vê o que é melhor. Claro que tem situações que tu simplesmente vai cumprir porque, sei lá… mas, tem muitos momentos em que você está extrapolando possibilidades. Pensar demais em um detalhe, pode tá antecipando uma discussão que seria melhor resolvida colocando um protótipo na mão do usuário final. Protótipos tendem a ser boas ferramentas para auxiliar, porque eles são bons o suficiente para implementar, mas eles não são completos o suficiente para virar regra imutável.

E: Eu não sei assim… Talvez esse lance não seja regra suficiente para ser imutável… Eu acho que isso acaba dependendo também do UX, né? O quanto você se agarra àquele material que você fez.
Dev 1: Verdade. Pode ter um pouco de pessoal também. O quanto que tu te apega. Exatamente.

E: Eh… Uma coisa que eu queria saber é se, tipo, você sente falta quando você pega esses materiais de UX, tipo, tela, mapa de fluxo, protótipo e tal… você às vezes sente que essa documentação às vezes ela é muito visual e você não tem uma noção clara do que é a arquitetura entre aquelas telas? Tipo, às vezes você sente falta de algum tipo de informação?
Dev 1: No contexto de vida com um todo, não necessariamente aqui, sim. Eu já tive situações onde a tela era linda, mas não ficava claro como que tu navegava entre elas e como é que o fluxo de telas fazia com que o usuário resolvesse o problema dele. E tive outras situações onde que não, tava lindo, maravilhoso e tudo era bem entendido super bem assim. Fazia super sentido o por que daquela informação estar naquela tela e não na outra. Para não antecipar uma decisão ou sei lá qual é o fluxo que tinha naquela versão.

E: Mas você consegue identificar possíveis razões ou características diferentes entre essas duas entregas para uma ser melhor do que a outra?
Dev 1: Eu diria que a principal diferença estava no entendimento de quem fez, que não é só o designer, é todo mundo. Se muito foi feito pelo designer, a responsabilidade maior cai sobre ele, não tem como fugir, mas muitas vezes, a formação da pessoa é mais de design visual… Não sei se esse é o termo correto, então, a pessoa trabalha bastante questões de tipografia, de cores e afins, mas se esquece de que não tem que ser só bonito, tem que ser funcional. Não é que a pessoa seja ruim, não é isso… Talvez ela não tenha sido exposta ao entendimento do que precisa, entende? Por isso, eu falei de entender.

Talvez ela não entenda que aquilo que ela fez é bom, mas que é apenas parte do todo e não o todo, e daí, de novo, vem a questão do apego: ela só sabe fazer aquilo e não está aberta a mudanças; porque para ela nem tem como haver uma mudança, aquilo já está perfeito.

E: Você já trabalhou com outros designers UX que tivessem uma formação diferente de design gráfico? Tipo jornalismo ou biblioteconomia? Foi diferente essa experiência que você teve com esses designers que tinham formação diferente?
Dev 1: Serve um UX que teve formação em computação?

E: Sim
Dev 1: Essas pessoas em algum momento da vida desenvolveram [código] e depois foram para área de design mais puro, digamos assim, me parecem ter um pouco mais de entendimento do todo. Talvez por eles já terem sofrido com outros designers que não tiveram isso antes. Porque tu te obriga a pensar na solução do problema como um todo e não só a parte dele, por isso, quando tu pensa em ux, tu pensa em design, tu pensa na experiência do usuário e não pensa só na beleza da tela, super simplificando.
[Ele comentou de um amigo nutricionista que fez uma cadeira de algoritmos]
Ele tem mais sistema num sentido sistêmico mesmo. Ele não programa nada, ele é nutricionista, mas ele diz que a própria forma dele abordar os problemas dele em nutrição, ele acha que é diferente de alguns colegas dele, por ele ter tido essa formação.

E: Existe alguma coisa que você queria que designers soubessem ou tivessem em mente enquanto estão produzindo esses materiais para vocês? Nem que seja como tornar isso mais rápido, mais ágil, mais completo…

Dev 1: Eu acho que os dois grandes problemas atuais que eu mais vejo são: Não fiquem separados do restante da empresa como um todo; tipo, qualquer um que entra em um silo, qualquer um mesmo, erra. Há uma diferença entre um grupo de pessoas que discute coisas e cresce de um grupo de pessoas que trabalha sozinha e acaba coisas e passa adiante. Isso é a maneira tradicional de fazer as coisas, projetizada e isso não serve mais para o século XXI. Isso serviu muito bem para o século XX, onde empresas e pessoas eram vistas como máquinas e isso tá mudando, graças a Deus, e não tem porque perpetuar esse tipo de *mindset*, isso não nos ajuda. E a segunda coisa é, não interessa o quão bonito for, se ninguém for lá e pegar o teu produto da prateleira, ele vai tá como se estivesse em um museu. Você fez arte e arte vai para o museu, se você fez um produto, você fez para alguém consumir este produto. Não esqueça nunca isso, né. Eu acho que eu posso estar chovendo no molhado, mas a impressão é de que às vezes eu acho que existe uma supervalorização ao belo e pouca valorização ao pragmatismo.

E: Mas esse pragmatismo, na sua opinião, ele tem a ver com a própria função que a tela tem?
Dev 1: Sim, certamente. É um ótimo termo. Uma tela tem uma função. E não interessa o quão bonita ela for, se ela não está alcançando aquela função, e, por isso que eu acho que, UX tem que se apropriar de métricas para saber se a função foi atingida ou não, ela não serve. Não interessa qual é o software, sabe? Tem uma função e o design tem que ser focado nessa função. Tipo, este pragmatismo tem relação com a própria função que a tela tem. Não interessa o quão bonita a tela for, se ela não estiver cumprindo a função, por isso, ux tem que se apropriar de métricas para descobrir se a função foi alcançada ou não, se não, ela não serve. Não interessa qual o software, sabe? Tem uma função e tem que ser focado nessa função.

E: Como é para você consumir as informações dessas telas? Depende muito do designer que faz, das ferramentas disponíveis…?

Dev 1: Depende de todas essas coisas. Tudo isso é o caldeirão de dificuldade da vida real assim… Acho que tem ferramentas que são melhores do que outras para entender as coisas, eu acho que tem designer que têm mais entendimento sobre as coisas e mais maneira de trabalhar focando em outras coisas do que outros. Isso é a graça que é viver e trabalhar com pessoas diferentes.

E: Você queria fazer algum comentário, alguma sugestão para mim?
Dev 1: Eu te diria que foi super tranquilo conversar contigo. Só deixar claro que é uma percepção de quem não é UX, então, talvez, eu tenha passado o limite de algumas coisas por simplesmente não entender. Então, se eu estiver errado, pode me dizer ai depois.... Sempre bom aprender e, se tu precisar trocar uma ideia de novo, é só entrar em contato.
E: Beleza! Muito obrigada. Valeu.
Dev 1: De nada! Valeu!

Dev 2

E: Poderia me dizer com o que você trabalha hoje e há quanto tempo você exerce essa função?
Dev 2: Com o que eu trabalho hoje? Eu trabalho como especialista de desenvolvimento de plataforma… Eu trabalho no Player[73] há uns 6 anos aproximadamente e com um time exclusivamente dedicado ao Player, há uns 4.

E: E quais são suas responsabilidades e etapas de trabalho?

[73] Equipe dedicada à execução de vídeos em páginas

Dev 2: A minha em particular, é manter a arquitetura e o Player consistentes entre plataformas, garantir o bom funcionamento dos vídeos nas plataformas e, para isso, temos bastante interface com os times de entrega de vídeo: CDM e afins, os times de produtos que usam o Player, que colocam os vídeos, e temos 3 times, meio que são o mesmo time, mas separados por plataformas distintas. E o modelo de trabalho que a gente usa é o mais ágil, a gente usa uma modalidade de SCRUM. Na verdade não é SCRUM puro, porque tem várias influências de Kanban e afins e, basicamente, o que a gente precisa usar dessas metodologias, a gente agrupa de acordo e a gente também tenta fazer isso na integração com os times.

E: Você poderia dizer quais são essas partes que compõem esse SCRUM que não é "puro"? Tipo, o que que tem de Kanban, o que que tem de SCRUM… Você consegue pensar em algum exemplo?
Dev 2: Por exemplo, no caso do Kanban, você tem alguns estados, você tem uma *track*[74], na qual se a tarefa está a ser feita, está em progresso ou se está feita… onde cada uma dessas etapas, a gente tem subdivisões que são derivadas do processo, mas é basicamente o mesmo que é "a fazer", "fazendo" e "feito". No caso, a gente tem algumas etapas de revisão de código e afins e… além disso, a gente impõe um grau de paralelismo que é para o time não abrir muitas frentes, não paralelizar muitas coisas e, ao mesmo tempo, como a gente tem um limite de tarefas sendo feitas. Quando a gente tem algo bloqueado, a gente tem que sinalizar isso imediatamente para que uma tarefa fique bloqueada o menos possível… Isso é uma das coisas do Kanban que a gente usa. No caso do SCRUM, a gente aproveita alguma coisas de ciclos, cerimônias e afins… não é 100% dos conceitos, porque o SCRUM, ele é um tanto quanto rígido e não se adapta tanto à flexibilidade que a gente gosta de exercer aqui.

E: Você consegue encaixar alguma outra metodologia no lugar do SCRUM, então? Tipo, que se assemelha mais?

[74] traquemanto

Dev 2: Então, a gente não encontrou nenhuma mais aberta e voltada para essa adaptação do Kanban que a gente usa, então, a gente nem chama de SCRUM na verdade, mas a gente atua como tal. A gente atua em sprints, mas, essencialmente, são ciclos… Não necessariamente, a gente estabelece uma meta específica por ciclo.

E: E… isso é uma visão de como desenvolvimento trabalha?
Dev 2: Sim

E: Do que...
Dev 2: Queria mencionar o caso de UX?
E: Isso
Dev 2: Aham, no caso de UX, o que que acontece é que UX entra sempre tanto na definição de uma tarefa ser feita, então muitas vezes, existe a necessidade de algo estar preparado por UX previamente, então, UX trabalha em ciclos similares ao desenvolvimento, mas um pouco antecipados, então, o insumo do que a gente usa para desenvolvimento vem dos sprints de UX, mas, ao mesmo tempo, UX também trabalha na parte de validação, planejamento, definição das tarefas de desenvolvimento.

E: É… posso falar uma coisa do fundo do meu coração?
Dev 2: Pode
E: Eu acho que UX trabalhar em SCRUM, é uma parada que eu não consigo tangibilizar na minha cabeça. Tipo, eu não consigo estar com desenvolvimento, tipo, os dois começarem do zero… Eu acho que isso nem rola e nem deveria rolar.
Dev 2: Então, até poderia rolar, mas, eh… ele exige um pouco mais de sincronismo no caso de terem objetivos; que as metas e objetivos tem que estar bem definidos para UX e *dev* entrarem ao mesmo tempo. Porque é muito difícil ficar em sprints exploratórios até você chegar em uma definição para depois seguir para desenvolvimento, mas também é muito útil do ponto de vista de prototipação.

Imagina você fazer o design em conjunto com os protótipos, então, suponha que se o designer tem as *skills* para prototipar e todas as ferramentas necessárias, *dev* não seria tão necessário nesse ponto.

E: Uma pergunta, desculpa te interromper, mas protótipo, você quer dizer seria tipo ferramenta de UX de prototipagem do tipo um *Protopie*, um *Principal* ou tipo, efetivamente código?
Dev 2: Ou efetivamente código. Poderia, por exemplo, ter algo rodando em um device. Uma coisa que eu tava discutindo com um colega, era poder demonstrar, poder fazer testes de usabilidade com usuários em laboratório. Eh… Muitas ferramentas dessas não conseguem executar bem código, especialmente tocando vídeo, então, esse tipo de protótipo, seria bastante útil ter um desenvolvedor trabalhando junto, porque aí a gente consegue ter algo realmente tangível, no sentido de, a sua aplicação funciona como deveria, tem os comportamentos esperados e não necessariamente ela ilustra o que deveria ser. Menos ilustração e mais uma demonstração.

E: Uma demonstração, mas uma demonstração assim já operante, né?
Dev 2: Sim, operante. Ela vai ser limitada, porém já operante, pra validar mesmo as suas hipóteses.

E: Por que já não existe uma ferramenta assim de UX?
Dev 2: Isso, na verdade, é um problema clássico de desenvolvimento. Porque, normalmente, existem modelagens, né, existem geradores de código que a gente usa pra, a partir do momento que a gente cria um desenho, que a gente cria uma especificação funcional, ela conseguiria extrair isso e gerar código executável para a gente, mesmo que ela não implemente mesmo a ação, mas que já implemente a, digamos, a forma da aplicação é bastante limitado. Por quê? Aí, tem a ver com expressividade. Tem algumas teorias de computação que [dizem] que quanto mais expressividade você tem em uma linguagem, mais ambiguidade você gera.

Para você eliminar as ambiguidades e conseguir gerar um código que entenda perfeitamente o que você tá fazendo é um problema bastante complexo, então, você precisa isolar esses tipos de problema. Então, como você precisa isolar, acaba que as suas ferramentas de prototipação e de especificação funcional mesmo, elas são limitadas propositalmente para você conseguir gerar algo que funcione. É muito difícil você gerar algo que funciona dado que você tem toda sua expressividade à disposição, então, as ferramentas, tendem a ser mais limitadas por isso.

E: Você pode explicar um pouco melhor o que seria expressividade?
Dev 2: É um pouco complicado mesmo, mas suponha que eu quero desenvolver um programa de computador que diz para ele, o que ele faz. A nossa linguagem é muito aberta, ela não é uma coisa preenchida em moldes, a quantidade de símbolos de uma linguagem necessários para expressar uma ação ou ideia é muito menor, porque você consegue, dado o contexto e afins, chegar a um resultado. Então, quando eu to falando, você não entende só o que eu to falando, você entende todo o contexto no qual a gente tá falando e você tem suas próprias memórias e experiências para se basear para poder compreender o que eu to falando. Num programa, você não tem isso. O contexto delas é puramente o próprio programa, então, tudo que o programa sabe tem que estar definido dentro dele.

E: Então, eu imagino que para vocês esse código que ferramentas de prototipagem dão então não deve ajudar em nada então, né?
Dev 2: É bem raso… Geralmente, o CSS que eles dão é bastante sintético. Acaba que ali tem muitas coisas que são desnecessárias, então, existem questões de otimização, existe questão de "será que isso atende o nosso propósito?" ou, ah, beleza, ele só funciona em determinada plataforma e por aí vai.

E: Qual você diria que é o papel do designer dentro do seu workflow? Como que a gente, UX, é útil para vocês?

Dev 2: Muitas coisas. Na verdade, eu acho que design, ele dá… ele me diz qual a melhor maneira de apresentar conteúdo ou de interagir com o usuário, ou seja, tem uma integração mais orgânica sobre como o usuário vai ter acesso a essa funcionalidade, então, ele vai definir a forma e o conteúdo do que eu preciso entregar para o usuário. Porque a programação em si só diz o que vai ser feito, agora o como vai ser feito, o como vai ser usado, o como vai interagir, é uma disciplina de design. É até estudado especialmente por quem estuda a parte de ciência da computação algumas coisas relacionadas à interação entre usuário e o programa, mas é muito mais voltado para entender… eh… como é que eu vou dizer… entender os modos de utilização ou as abstrações de comportamento que a gente tem disponíveis para apresentar para o usuário, mas o como, a forma como as coisas devem ser feitas, ou a forma como o usuário vai interagir é algo que é completamente design. Não é uma coisa assim direto da implementação.

E: E como é que o design se enquadra dentro do seu workflow?
Dev 2: Bem, eu acho que é muito, como eu tinha falado antes, é uma coisa de definição. Ele vai trabalhar tanto no início de tudo quando as metas estão sendo definidas até o momento em que, quando a gente chega em uma conclusão de algo que vai ser desenvolvido, a definição de como e o que tem que ser apresentado, assim como, durante o desenvolvimento, a gente valida e verifica hipóteses. Então, a gente define hipóteses, define métricas e define resultados esperados, e a partir daí, a gente consegue extrair as nossas especificações para o sistema ser desenvolvido e depois validado. Então, o designer ele atua em todas as etapas, tanto antes quanto durante e depois [do desenvolvimento].

E: Que tipo de informações você precisa ter para fazer o seu trabalho? Quais são dados importantes que UX fornece para você?
Dev 2: Eu diria que o mais importante seria o comportamento do sistema em relação ao usuário. Digamos, o que o usuário é capaz de usar?; o que o usuário é capaz de fazer?; como ele deve interagir?. Então, o modelo de interação, o modelo de apresentação. Eu acho que essa seria a parte mais importante dessas definições.

E: Isso quer dizer que, por exemplo… um protótipo interativo? Ou não?
Dev 2: Protótipo interativo é uma possibilidade. Como a gente tinha mencionado antes em ferramentas de prototipação… Eu tenho algo que eu preciso apresentar, então, a estrutura de apresentação é um material importante que contém definições importantes para a gente.

E: Desculpa, eu te interromper, mas é para eu entender melhor. Estrutura de apresentação é tipo layout? Eu to pensando muito em coisas concretas…
Dev 2: É você tá pensando muito no concreto, eu to pensando mais em coisas abstratas… Tudo bem, um wireframe, um layout, protótipo… eh… todas essas formas são importantes, porque elas concretizam de certa forma a ideia que a gente tem de sistema, de alguma parte do sistema que a gente tem que entregar. Porque, assim, no momento em que a gente tem definições, a gente ainda não sabe muito ainda o que a gente vai fazer. Uma vez que você não tem um guideline, uma vez que você não tem um material de apresentação, de como aquilo precisa ser exibido para o usuário ou que formas de interação você oferece ao seu usuário, a partir do momento em que você deixa isso em aberto, a implementação do sistema pode ser feita da mesma forma. Eu posso ter infinitos modelos de interação e aí, essa parte "livre", ela gera muitos problemas… Lembra? Ambiguidade, incerteza, sempre gera muitos problemas em programação, porque o software, ele não pode ser ambíguo, ele não pode ser livre. Ele, na verdade, tem que ter uma quantidade de operações, uma capacidade bem delimitada para ele funcionar bem, se não, quando você foge dessas delimitações, tudo fica imprevisível. Então, ajudar a concretizar tanto a capacidade e operações e formas de interações do sistema assim como limitações, qual o escopo do que a gente tem que interagir para a gente poder cercar, proteger de comportamentos inesperados, tudo isso é importante, a gente ter esse material que vem do design.

E: Entre os materiais que você recebe hoje, você considera que essas informações todas estão contempladas neles? É uma documentação completa?

Dev 2: Hoje, eu acho que não. Porque eu acho que, assim, muitos dos padrões de interação, inclusive, volta e meia quando a gente questiona de onde eles vieram, às vezes não se sabe, porque realmente não existe uma documentação concretizada daquilo. É só o que foi passado de dev para dev, de designer para designer e por aí vai. Então parte dessas documentações são coisas que a gente tá tentando reescrever ou redefinir, exatamente para a gente ter o conjunto de documentação completo.

E: Geralmente que materiais você recebe de UX?
Dev 2: Bem, normalmente, temos alguns wireframes, algumas imagens, temos protótipo… protótipo estático no Zeplin com as especificações visuais menos [as] de interação… eventualmente tem também protótipo do Principal ou Overflow. Coisas [ferramentas] que tentam fazer tudo não funcionam direito.

E: Ah, lembrei o que eu ia perguntar… Nesse sentido de guidelines bem definidas, é por isso que o Material é tão maravilhoso tanto para designers quanto para desenvolvedores?
Dev 2: Sim, realmente é, porque, o que acontece, o Material tem interações bem definidas, por causa disso, você consegue escrever código que implementa esses guidelines e compartilhar entre aplicações.

E: Você tem algum *pain point* específico com relação à documentação? Porque você falou que, de repente, as informações não estão completas… Mas, além desse pain point, existe algum outro?
Dev 2: Na verdade, um dos *pain points* principais é a informação não estar centralizada. Às vezes é complicado, porque, às vezes um comportamento tá definido em uma certa aplicação em uma ferramenta, mas a apresentação está definida em outra e agregar tudo isso é meio complicado. Manter um registro de toda documentação que existe é complicado, é um *pain point*. Óbvio que a ausência da documentação também, como eu tinha dito antes.

E: Mas por documentação, por exemplo, você entende o que?

Dev 2: Quando eu penso em documentação, é a consolidação desses materiais, porque uma vez que a gente determina… porque, por exemplo, uma vez que eu conversei com você, a gente desenhou no quadro alguma solução. Isso é um primeiro registro, tiramos uma foto, guardamos, a gente tem isso compartilhado. O material vai evoluindo, a gente chega a um *sketch*. Todos esses passos, de registro das nossas ideias e dos nossos acordos, é o que eu compreendo como documentação. Então, a documentação, ela pode ser um protótipo, ela pode ser um documento, ela pode ser uma imagem, ela pode ser várias coisas. Eu acho que o conjunto disso tudo formaria a documentação.

E: É o que está consolidado então. No momento em que uma coisa se torna obsoleta, ela não faz mais parte da documentação.
Dev 2: Não necessariamente. O histórico faz parte da documentação, porque isso ajuda a entender qual foi o caminho e quais foram as decisões tomadas e por quê… Por exemplo, se a gente quiser voltar atrás em uma decisão, a gente consegue reavaliar ou então, realmente, a gente descobre que não precisa reavaliar, que a gente já experimentou isso. Então, ter o histórico é importante. Não é só a consolidação daquele momento. É a consolidação de tudo: tanto o histórico quanto o atual.

E: Mas o histórico então, ele só teria valor para consultas muito pontuais, por exemplo? Ninguém fica lendo histórico...
Dev 2: Sim, o histórico, ele é exatamente isso. Ele é o histórico de decisões passadas e afins.

E: Como essa documentação chega geralmente até você? Possui alguma ferramenta específica?

Dev 2: Normalmente, a gente tem contas ou um repositório compartilhado... para guardar esses arquivos e informações ou uma conta em um serviço de nuvem. Por exemplo, o Zeplin é um deles, mas os arquivos em si, a gente tem o Google Drive, a gente tem o Filer aqui que a gente usa para compartilhar a estrutura de diretórios onde estão os arquivos para serem compartilhados. Não é uma coisa simples de se consultar históricos, a gente não consegue definir relações específicas, mas, novamente, uma vez que alguém tem o contexto daquela solução, é mais fácil explorar esse conjunto de documentação.

E: E uma pergunta, nesse sentido de dar contexto, de consultar material e tudo mais, você tende a achar que… qual a granularidade que a informação deve ter? Tipo, você imagina que você veria mais valor em uma coisa muito assertiva do tipo "este botão mudou" ou, por exemplo, apresentar a interface toda de novo para você? Você acha melhor ter só a feature ou todo o contexto da tela?
Dev 2: Na verdade é uma resposta difícil, porque varia de caso a caso… Muitas vezes ter o que mudou, só o que mudou ali, é mais importante, porque você tem o foco daquele ponto, mas você perde o panorama geral… Então… Eu acho que a gente fica limitado pelas ferramentas, mas o ideal seria a gente ter ambos. Da mesma forma que a gente faz no código. No código, a gente sempre consegue enxergar a diferença, ou seja, o que foi mudado passo-a-passo a cada *commit*[75], a gente tem a diferença, mas, uma vez que você faz *checkout*[76] daquele *commit*, você tem exatamente tudo o que seria daquele ponto. Então, você tem o panorama geral, mas se você quiser consultar a diferença, você consegue. Eu acho que ambos são importantes, porque a diferença é interessante para o momento em que a gente está fazendo uma nova mudança; só a diferença, o que mudou, é importante para aquele momento, mas em algum momento que a gente queira um marco, pode não ser a qualquer momento, porque não é tão granular assim.

E: E essas ferramentas satisfazem as suas necessidades? Elas são mal usadas?

[75] Fazer upload ou o envio de código no banco de dados
[76] *Checkout* de um *commit* é o ato de atualizar o código com as alterações vindas pelo *commit*

Dev 2: Eu acho que… é difícil, porque normalmente a gente não consegue documentar tudo, muitas coisas ficam implícitas ainda. Talvez quando a gente chegar em um ponto que todas as interações estejam documentadas, talvez não seja [mais] um *pain point*, mas, até lá, eu acho que ainda é um pouco. Mas as ferramentas em si, elas são… elas atendem ao propósito individual de cada uma, o problema é: todo esse conjunto[77] não é o ideal. Talvez algo que agregasse mais informação fosse mais simples. Tipo, facilitar a consulta, por exemplo, ou facilitar exatamente essa visualização que mudou de antes para agora; conseguir comparar o antes e o depois e só ver as diferenças… isso também é interessante. Eu acho que, assim, ainda tem muito para evoluir nesse sentido, inclusive a questão dos protótipos. O protótipo, ele é um material muito mais pontual, ele não tá presente em tudo. Então, muitas vezes, a descrição de como as coisas funcionam, a visualização de como o produto trabalha não é tão simples, ele requer interpretação. Talvez essa seja uma possível melhoria.

E: Eh… agora, uma pergunta que tem até relação com o *mindset* que o design tem. Eu fico em dúvida sobre como acontece o faseamento para desenvolvimento. As fases e a priorização para se chegar ao produto final.
Dev 2: Qual o problema do faseamento? Para desenvolvimento, é muito mais fácil, a gente fasear, porque a gente consegue pensar em componentes e em quais partes a gente consegue desenvolver para que elas funcionem sozinhas, mas, para funcionar sozinhas, não quer dizer que elas façam sentido. Então, para desenvolvimento, os blocos, eles podem funcionar sem fazer sentido para o usuário. Então, é mais fácil interpretar um MVP do ponto de vista de desenvolvimento do que do ponto de vista de design, certo? Mas dito isso, o que acontece em geral. Como que funciona o desenvolvimento: a gente sempre modulariza, a gente sempre cria blocos da aplicação. Blocos lógicos, blocos que processam coisas que entregam *output*[78] e blocos de apresentação, então, é meio que um conjunto de lego, a gente vai encaixando. É modular e mais simples de tratar, mas do ponto de vista de design, às vezes, isso não faz sentido.

[77] Imagino que ele quis dizer conjunto de ferramentas
[78] Traduzido como saída. No contexto, refere-se a entrega de feedback

E: É porque eu acho que o *mindset* de design é meio contraditório com o *mindset* ágil. Eu acho que a gente em design [a gente] tem muita dificuldade com isso, porque a gente já pensa no final.
Dev 2: Eu acho, e aí já é "achismo", eu tenho uma opinião… Talvez a gente possa trocar uma ideia entre designers e desenvolvedores para entender como que desenvolvimento funciona para tentar pensar em estágios intermediários, porque eu acho que o problema é que, em geral, a gente pensa no produto final. Ai, beleza, o produto final tá aqui, mas eu vou levar 3 meses e eu preciso entregar a primeira coisa daqui a duas semanas e eu só posso entregar um tanto de coisas até esse prazo, então, talvez esse faseamento do desenvolvimento pudesse ser mais sincronizado com o que designer… tem que acompanhar. Beleza, qual a primeira fase, qual que é o mínimo? Essas funcionalidades, então, vamos desenhar juntos. A gente já consegue criar algo de apresentação para esse conjunto de funcionalidades para que o conjunto de componentes seja adaptável e evolua com o tempo.

E: Mas sob essa perspectiva de faseamento, como é que ficam as informações?
Dev 2: Chegamos a um acordo de que o MVP é esse e essas são as características essenciais da entrega. Se isso não for feito, esse MVP não deveria ser considerado entregue. Se o MVP não contempla essas características… o conceito de "pronto" tem que ser atender isso… Suponha, ela [a entrega] lá no nosso Kanban, se ela não passa pela etapa de "Validação", ela nunca vai chegar no "*Done*", ela nunca vai chegar no "Feito". Ela não tá feita ainda, tá em processo de validação.

E: Mas sob a perspectiva disso [a acumulação de features não entregues] ir crescendo… Por exemplo, tinham 5 features, fizeram 2… Tipo, quando é que essas 2 são consideradas "*Done*" para poder seguir com as outras.... Porque eu sempre penso em *Agile* como algo que a gente meio que revisita assim…

Dev 2: É o lance dos ciclos, é o ciclo que determina isso. Então, assim, "Ah, no primeiro ciclo, eu vou entregar as primeiras *features*", ai, entreguei, mas elas ainda não estão prontas, tem algum detalhe faltando. O próximo ciclo de interação deveria revisitar elas. "Ah, beleza, eu tenho essa funcionalidade, mas eu tenho que revisitar essas duas para corrigir ou ajustar ou acrescentar coisas que ficaram faltando da entrega anterior"... É porque, assim, o processo ágil, ele é interativo e incremental. Não é interativo, porque eu entreguei. A cada interação, eu tenho que ver o que tá entregue e o que deveria ser entregue e seguir adiante. Então, assim, eu construo em cima do que foi feito antes, mas eu vou incrementando… Eu vou, eh, acrescentando novas *features* ou eu vou acrescentando detalhes ou eu vou acrescentando visualização ou acrescentando qualquer coisa que a gente determine que precisa ser incrementada naquele ponto. Então são novas funcionalidades e incrementos. A gente sempre revisita. Às vezes acontece daquele grupo de *features* não estar priorizado, isso é um problema quando você tem um time que cuida de muitas coisas ao mesmo tempo. Então, assim, você vê que o time está sobrecarregado. Isso é um sintoma, mas, o fato de revisitar ou não, não é muito por que isso foi despriorizado, mas sim porque outras coisas surgiram na frente. Agora, revisitar aquilo é algo que vai acontecer eventualmente. É diferente de "Ah, beleza, to aqui no desenvolvimento, tenho 5 funcionalidades, criei duas, nessas duas, ficaram pendentes alguns detalhes, eu vou completar as cinco e não vou revisitar mais.", não, enquanto, eu estou trabalhando nesse mesmo conjunto de funcionalidades, nesse mesmo produto, eu tenho que voltar e revisitar a completude de tudo que eu tô entregando. Então, é parte desse ciclo de validação. Uma das coisas que a gente tem que faz parte de rigidez do SCRUM, é eu faço o planejamento; quando eu termino o meu ciclo, eu reviso o que foi feito, se tá de acordo com o meu planejamento. Se não está ou o que quer que seja, eu faço uma retrospectiva, eu faço uma análise do que contribuiu ou prejudicou as minhas entregas e quais as ações que eu preciso tomar. A partir do momento em que eu defino isso, eu continuo o meu ciclo, eu volto para o início. Então, eu sempre planejo, valido e verifico ações a serem tomadas. É um ciclo.

E: Eu ia perguntar, tipo, o que você diria que é a documentação final. É a documentação de UX ou o código? E daí, eu acho que, no final, isso depende né. Depende do projeto, depende do estágio em que ele está.
Dev 2: Depende

E: Existe alguma coisa que você gostaria que designers soubessem ou tivessem em mente quando a gente tá produzindo esses materiais?
Dev 2: Bom, acho que uma das coisas é o que a gente acabou de discutir de… eh… faseamento. Realmente, fasear é difícil. Tipo, em desenvolvimento já é difícil. Então, muitas vezes, assim, você vê discussões enormes, porque eu preciso fechar uma entrega com algum conjunto de funcionalidades, mas essa minha entrega não contempla um todo que foi desenhado. E ai, eh, a apresentação de pedaços, né, de parciais das minhas entregas é bastante complicado. Então, assim, talvez para a gente discutir quais abstrações existem e quais abstrações a gente quer aplicar sobre as nossas fases, talvez isso seja interessante para um designer entender, mesmo que ele não conheça exatamente os conceitos, pelo menos, assim, entender essa modularização, essa granularidade do desenvolvimento ajude um pouco a pensar em quais seriam as fases de entrega até mesmo para elaborar mesmo, criar uma entrega consistente.

E: O que, na sua opinião, essa metodologia, essa maneira de trabalho, o que ela tem de bom, o que ela tem de ruim?
Dev 2: O que eu acho que ela tem de bom é a possibilidade de você conseguir detectar desvios, problemas, etc, rapidamente e ajustar o curso. Então, quanto mais curto o seu ciclo, mais fácil de você manter isso, inclusive. Supondo… um ciclo de uma semana ou de duas semanas, a menos que seja algo extremamente grave, a cada duas semanas, [é mais fácil de] você conseguir detectar possíveis falhas no planejamento, pequenos problemas ou detalhes que faltem, que você não tenha conseguido encaixar no meio de caminho, pelo menos você tem ciclos de reavaliação, de feedback, muito rápidos, então essa é a principal vantagem que eu vejo no SCRUM e várias outras metodologias ágeis que usam esse modelo.

Mas no caso do SCRUM em particular, eu acho que existe uma certa rigidez no formato e, assim, muitas vezes acaba que, por questões de interpretação inclusive, que o SCRUM é uma metodologia aberta. As cerimônias que são as reuniões e aquelas reuniões que eu mencionei do ciclo, o *planning*, *review*, retrospectiva e afins, às vezes, elas tomam muito tempo ou, se você escolhe um ciclo muito curto, elas podem ser muito constantes. Então, é muito importante se policiar para evitar que isso gere um *overhead*[79] muito grande e desnecessário no seu desenvolvimento e é muito comum acontecer isso.

E: Mas, ao mesmo tempo, é meio difícil, né… Eu sei que às vezes, a gente tá trabalhando e a gente é interrompido no *flow* para ter que ir em uma reunião, mas... Eu não sei como a gente conseguiria fugir disso…
Dev 2: Mas, não é nem interrupção. Uma coisa é: Se sua quebra no ritmo ai é planejada, por exemplo, suponha que toda sexta às três horas da tarde é a sua reunião. Você fez a sua reunião de *review*, você fez sua retrospectiva e tal, e aí, você quer preparar para segunda-feira um material para o *planning*… Repara, você tem um pequeno intervalo entre essas reuniões… Geralmente, o ideal é que *review* e retrospectiva aconteçam juntos e daí, você já tem o seu plano de ação. E, beleza, digamos que você chegou no fim do seu dia, mas na segunda-feira, logo pela manhã, você tem a sua reunião, então o seu *flow*, supondo que seja de segunda a sexta o seu ciclo, você teria o restante da semana inteiro, teoricamente, sem interrupções e as reuniões que envolvem coisas para dar insumos para esse planejamento ou pra *review* e etc, eles podem estar mais próximos dessas reuniões. Você não deveria ter tantas quebras de ritmo a menos que fosse algo emergencial. Eu acho que o problema maior é a extensão que as coisas tomam.

E: Às vezes as coisas entram em uma especificidade, né…
Dev 2: É muito difícil se policiar nesse sentido.

[79] Em ciência da computação, é qualquer processamento ou armazenamento em excesso de um recurso.

E: E aí para documentação de UX isso é muito difícil, porque a gente tem que ser muito específico e esse nível de detalhe pode não ser percebido e aí, a gente tem que ficar "Olha, é assim dessa maneira"
Dev 2: Eu acho que assim, especialmente documentação de UX tem que ter uma granularidade variável. Do tipo, ah, coisas que são em linhas gerais para apresentar *flow* e etc, não precisam entrar tanto em detalhes, mas quando você chega em uma especificação de interação, interface, etc, eu acho que aí já é mais necessário entrar em detalhes até por que é bom ter essa documentação para cada um ficar ciente daquela funcionalidade em particular, porque, se você deixar em aberto, a comunicação fica verbal e aÍ a comunicação verbal fica muito entre somente as pessoas que estão envolvidas e não o todo, né, o time, por exemplo.

E: Uma pergunta, uma que um desenvolvedor que eu conversei há um tempo atrás, o [Dev 1], ele falava muito de UX trabalhar junto ao mesmo tempo mesmo com desenvolvimento. O meu ponto de tudo isso era para dizer se vocês de desenvolvimento sentem que tudo chega muito depois do trabalho de UX. Do tipo, UX começa uma parada e só eras depois vocês vão ter acesso a isso.
Dev 2: Então, isso é um problema nosso que é em geral é: Essa relação fica um tanto *Waterfall*. Dado que a gente tem alguma coisa para fazer, a gente segue um ciclo modelo ágil, até chegar nessa coisa, até chegar na definição dessa coisa, muita coisa já rolou, muitas outras pessoas e ou times foram envolvidos, inclusive outros UX e só o resultado final chega para o dev. E muitas vezes a gente acaba tendo que validar o modelo e tal e pode não ser exatamente o caminho que a gente queria… O resultado não é tão bom quanto o esperado, o resultado não é exatamente o esperado e a gente precisa voltar. Esse ciclo fica muito longo. Então, talvez, aproximar isso fosse melhor. Um dos sonhos, o modelo ideal seria, ah, beleza, pessoal de UX vai desenhar, vai documentar, etc, mas já vai prototipando, mesmo que não seja a versão final. Prototipar a interação, prototipar como vai ser aquela estrutura, como vai ser o fluxo, navegação, etc, porque certas respostas que viriam mais tarde, poderiam vir ali no início, nesse início de implementação. Nesse início, eu acho que faz sentido ter um par *UX-Dev,* por exemplo, ou quando a gente quer validar algo mais lá na frente mesmo.

E: Só para eu entender melhor, nessa parte de implementação, você já quer dizer esse sonho dourado dos protótipos já serem reais, né? Não um *Protopie*[80] que só movimenta, animação… "Código-código".
Dev 2: Isso. Mesmo que não seja o código final. Mesmo que não esteja otimizado, assim, o ideal para funcionar ou que ele precise ser convertido para outra tecnologia, esses protótipos, eles são úteis para facilitar o entendimento. Eles servem inclusive para documentação, mas ao mesmo tempo, eles servem para aproximar um pouco *UX* de *Dev,* porque as limitações de desenvolvimento que às vezes a gente tem dificuldade de mostrar e ilustrar por que que tal botão não vai conseguir estar perfeitamente alinhado ou porque a gente precisa aumentar alguma área de toque de interação, etc, porque, na prática, aquele código não vai funcionar, a gente consegue chegar nessas conclusões mais rápido e isso, inclusive, aproxima um pouco a dor dos *devs* para o pessoal de *UX* e vice-versa.

E: Mas, então, você acha que, seria melhor começar a envolver vocês logo antes da gente já ter tela pronta.
Dev 2: É quando você já tem um esboço, um *wireframe*. Poderia prototipar o *wireframe*.

E: Isso foi uma coisa que o [Dev 1] comentou também. E nesse protótipo de *wireframe*, mesmo que não fosse assim… mesmo que fosse de *wireframe*, qual que seria o valor dele? O que que ele entregaria de legal para você?
Dev 2: Eu acho que assim, em parte, ele vai entregar um tipo de especificação, mas também ele vai aproximar, vai cobrir um pouco esse *gap*[81] do tipo: *UX* pensou nessa estrutura, nesse modelo e isso ainda vai refletir no visual antes de chegar no *dev*. Se o *dev* tá ali, ele já vai, ah, beleza, a gente já vai preparar essa estrutura, eu consigo modelar algo para atender aquilo. Facilita um pouco a modelagem sabendo a priori o que vai ser feito e qual é a estrutura mesmo que a gente ainda não tenha as animações e essa parte visual. Só da gente conseguir enxergar esse ponto do design, a gente já consegue também planejar a estrutura correspondente, então,

[80] Software de prototipação
[81] Lacuna

isso poderia servir de documentação para *UX* e para *dev*, inclusive para quem vai dar manutenção para o código.

E: Existe mais alguma outra dificuldade que você encontra de lidar com *UX* dentro desse modelo de trabalho?
Dev 2: Eu não sei… Eu acho que tudo acaba remetendo ao distanciamento. Seja temporal, das coisas serem desenhadas muito antes de ser começada a implementação quanto esse afastamento assim do pessoal de desenvolvimento não entender completamente como funciona a dinâmica de *UX* e vice-versa. Mas, assim, não vejo muitos outros problemas. Vejo mais ramificações desse distanciamento. Eu acho que essa aproximação maior, que ela resolveria bastante problema e não é uma aproximação só de "Pô, vem trabalhar comigo", é realmente a questão de compreensão, facilita muito compreensão. Às vezes, o pessoal de desenvolvimento quer só entregar e não importa exatamente o que, quer só que funcione e não entende que, o problema é que, não tá entregando algo só funcional… O problema quando a gente entrega uma interface para o usuário é que não é só função, não é só o que aquilo faz, mas é que informação tá sendo trocada ali também. A informação, a apresentação faz parte da implementação. Quanto mais tempo você leva para rever o processo, mais acúmulos de débitos técnicos, pendências e afins você vai ter que pagar.

E: Em *Agile*, eu entendo que para não lidar ou, pelo menos, para minimizar essas pendências que a gente ou começa tendo uma mente super clara de como isso vai se desenvolver para você conseguir ter a base de tudo isso e para conseguir construir em cima ou, então, inevitavelmente você se perde.
Dev 2: Então, eu não concordo bem com isso, mas existe essa impressão mesmo. Vamos lá. Suponha que você tá completamente alinhado, todo mundo sabe já completamente o que tem que fazer, as incógnitas estão controladas, o time tem um certo grau de relaxamento pra poder seguir adiante. Então, a cada ciclo tem uma folga, você consegue acomodar variações no seu desenvolvimento, no seu planejamento, e aí, beleza. Tudo funciona. Isso é o que muita gente acha do modelo ágil, o que não é verdade. Existe prazo, existe correria, coisas mal feitas…

O que que acontece com modelo ágil no geral é que pessoas associam muito isso a entregar rápido, e ágil não é rápido, é sinônimo de tempo de resposta. Eu consigo responder rapidamente à mudanças, à estímulos, qualquer variação. Então assim, você vai entregar, você vai descobrir que tem erros, você vai entregar, vai descobrir que faltou alguma coisa ou você vai ter pressão de prazo e vai ter que entregar algo que está aquém do planejado, seja tecnicamente ou seja de *UX*, mas o importante é que você faça um planejamento para que se a sua decisão aqui impõe um débito técnico, algo que assim, olha, se eu não conseguir implementar esse botão novo, não vou conseguir fazer tal coisa ou testei mal isso aqui, não consegui fazer nenhum teste especial ou validação manual. O ponto é, tudo que você fez aqui tem que ser pago em um prazo finito de tempo, porque se você não paga dentro desse seu prazo, essencialmente, você está destruindo o seu projeto, porque chega em um ponto que aquilo ali vai cobrar, vai cobrar o preço. E aí o pagamento pode ser, pagar bem caro ali na frente, ou, ah, esquece, não consigo fazer isso aqui, você fazer do zero, por isso que, muitas vezes, os projetos morrem e começam do zero de novo. E é sempre assim, porque existe essa associação de que agilidade… Entendeu? Agilidade não é entregar rápido. É que existe uma visão distorcida com relação a isso. Muitas vezes os times criam débitos técnicos sem que na verdade, eles tenham um planejamento pra amortizar esse débito. A gente tem débitos de tempos que vão entrando aos poucos, mas, no geral, a gente paga os débitos de *UX* antes de pagar os débitos técnicos, porque, geralmente o débito técnico, ele impõe mais dificuldade de manutenção, evolução, etc, mas ele é menos perceptível, mas se você entrega uma experiência ruim para o usuário, você acaba… eh…. Você precisa consertar isso antes, mais rápido. Tá mais explícito, tá mais na cara do usuário. *Agile* não é só uma metodologia, ele tem uma filosofia por trás… Talvez seja esse o problema de compreensão.

E: Se você pudesse alterar três coisas sobre como documentação de design é feita hoje, o que você alteraria?

Dev 2: Eu acho que a primeira, seria a questão do protótipo, que assim, a maior parte das interações, estruturas de navegação mais complexas, integram o protótipo para, pelo menos, entender a sequência das coisas facilitaria, porque diminuiria bastante a comunicação verbal, a dependência de todos os envolvidos estarem nessa comunicação verbal e memorizarem isso. Outra coisa é, quando uma ideia surge, analisar a viabilidade técnica é interessante, daí, a gente discute limitações ou faz um protótipo e descobre que não é possível, mas não é possível pelo protótipo ou pela tecnologia? Então, bater com essas limitações tem que gerar esse estalo, deixar eu chegar fatos, porque muitas você acaba pensando muito na solução e dá de cara na implementação e o tempo que você gastou projetando aquilo… na verdade não é totalmente desperdiçado, porque você explorou uma ideia, mas é algo que não é concretizável… não é usável. Tem algo que é, beleza, isso aqui foi um exercício de abstração, puramente de abstração, e ai, ter mais ciência disso, diminui o peso desses caminhos… porque, ah, é um momento que eu posso abstrair aqui a vontade ou é um momento em que eu tenho que botar o pé no chão e seguir em frente com as limitações?

E: Mas isso é uma coisa que me deixa encucada com *Agile*… no manifesto *Agile*, ele comentam sobre "*smoochy stuff*"[82], tipo, coisas meio piegas e pegajosas, porque exige realmente que você tenha… Exige não, mas ele pelo menos, propõe que você tenha um alinhamento de valores e princípios enquanto equipe e não necessariamente isso acontece né… Tipo, você não pode esperar que todas as pessoas tenham a mesma índole, moral, mesmo valor ou o mesmo pique que você…

Dev 2: Mas é porque não é muito entre os integrantes… É um alinhamento assim, vamos chegar a um consenso. Então todos do time chegam ao consenso que é quais são os valores daquele time, qual é a missão, qual é… é como se fosse uma micro empresa. Vamos pensar quais seriam os valores daquele grupo, daquele pequeno conjunto. Não é muito voltado ao que você indivíduo tem por valores. É mais assim, pessoal, nós enquanto time que faz produto X, temos que valores?

[82] No manifesto ágil, é usado como sinônimo para "coisa brega e sentimental"

O que que é importante para X? O que é relevante? O que a gente acha que o usuário vai enxergar de valor em X? Qual é a nossa visão de *UX*, não só enquanto produto, mas enquanto, unidade, processo, etc. Como grupos externos a nós enxergam o nosso grupo? O que a gente quer passar para eles? Seriam mais esses princípios e não necessariamente questões individuais. O que essa causa representa? Quais são os valores dessa causa?

E: Você tem mais algum comentário para fazer?
Dev 2: Não, só isso.